A Hierarchical On-Line Path Planning Scheme using Wavelets

A Thesis
Presented to the Academic Faculty

by

Efstathios Bakolas

In partial fulfillment
of the requirements for the degree of
Master of Science
in Aerospace Engineering

School of Aerospace Engineering
Atlanta, Georgia 30332-0150 U.S.A.
May, 2007

A Hierarchical On-Line Path Planning Scheme using Wavelets

Approved by:

Dr. Panagiotis Tsiotras (Advisor)
Aerospace Engineering
Georgia Institute of Technology

Dr. Eric Feron
Aerospace Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: March 26, 2007

Epigraph

"*... instead of the great number of precepts of which logic is composed, I believed that the four following would prove perfectly sufficient for me, provided I took the firm and unwavering resolution never in a single instance to fail in observing them.*

The first was never to accept anything for true which I did not clearly know to be such; that is to say, carefully to avoid precipitancy and prejudice, and to comprise nothing more in my judgement than what was presented to my mind so clearly and distinctly as to exclude all ground of doubt.

The second, to divide each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution.

The third, to conduct my thoughts in such order that, by commencing with objects the simplest and easiest to know, I might ascend by little and little, and, as it were, step by step, to the knowledge of the more complex; assigning in thought a certain order even to those objects which in their own nature do not stand in a relation of antecedence and sequence.

And the last, in every case to make enumerations so complete, and reviews so general, that I might be assured that nothing was omitted."

"Discourse on the Method of Rightly Conducting One's Reason and of Seeking Truth in the Sciences",
Descartes René,
translation of Discours de la méthode,
Gutenberg Project at http://www.gutenberg.org.

Acknowledgements

I wish to express my appreciation to Dr. P. Tsiotras, Dr. M. Egerstedt and Dr. E. Feron, members of my Master Thesis committee, for their true interest in evaluating this work.

Furthermore, I would like to thank the A. Onassis Public Benefit Foundation, of which I have been a scholarship recipient for the last two years. This research work has been supported in part by NSF (award no. CMS-0510259) and ARO (award no. W911NF-05-1-0331). Their financial support is greatly appreciated.

Finally, I owe the greatest debt of gratitude to my family for the moral and emotional support they have provided me, especially during the years of my academic studies. Their affluent help has been a strong motivation and source of inspiration for continuing to fight for my ideals and ambitions. For these reasons I dedicate this work to them as the least token of appreciation.

Contents

List of Tables

List of Figures

Summary

The main objective of this thesis is to present a new path planning scheme for solving the shortest (collision-free) path problem for an agent (vehicle) operating in a partially known environment. We present two novel algorithms to solve the planning problem. For both of these approaches we assume that the agent has detailed knowledge of the environment and the obstacles only in the vicinity of its current position. Far away obstacles or the final destination are only partially known and may even change dynamically at each instant of time. The path planning scheme is based on information gathered on-line by the available on-board sensor devices. The solution minimizes the total length of the path with respect to a metric that includes actual path length, along with a risk-induced metric. In order to obtain an approximation of the whole configuration space at different levels of fidelity we use a wavelet approximation scheme. In the first proposed algorithm, the path-planning problem is solved using a multi-resolution cell decomposition of the environment obtained from the wavelet transform. In the second algorithm, we extend the results of the the first one by using the multiresolution representation of the environment in conjunction with a conformal mapping to polar coordinates. By performing the cell decomposition in polar coordinates, we can naturally incorporate sector-like cells that are adapted to the data representation collected by the on-board sensor devices.

Our approach is motivated by many typical navigation problems for autonomous vehicles, where a plethora of sensory devices used (e.g., cameras, radars, laser scanners, satellite imagery) usually have different ranges and resolutions. The proposed algorithms provide a computationally efficient path planning scheme which is able to combine the information provided by all the sensors in such a way that the computational resources are used on that part of the path (spatial and temporal) that needs them most.

The proposed planning scheme takes full advantage of any local information around the agent's current position. This allows the construction of a directed weighted graph of the approximated free space, the dimension of which is adapted to the on-board computational resources. By searching this graph we find the desired shortest path to the final destination using the Dijkstra's algorithm, provided that such a path exists. The path is a finite ordered sequence of visiting points corresponding to a polygonal line connecting the initial and goal destination. When dynamic constraints on the agent's motion are taken into consideration this polygonal line may be dynamically infeasible. Therefore, we introduce an *on-line* scheme which generates a collision-free, smooth trajectory. The trajectory passes sufficiently close to the points of the path at specified instants of time (time scheduling). For this scheme we employ a kinematic ground vehicle model to describe the agent's equations of motion and then apply some ideas of the input/output linearization theory (IOL) for MIMO systems to accomplish the trajectory generation and time scheduling tasks.

Finally, several simulations are presented to test the efficiency of the proposed path planning and trajectory generation schemes using non-trivial scenarios.

Chapter 1

Introduction

1.1 Introduction

The path planning problem has been under intense investigation for many years in several research areas, ranging from operation research, vehicle navigation, network optimization, etc. In the area of the autonomous vehicle navigation a major distinction between different path planning problems is based on whether the information about the operating environment is known *a priori* or is updated with time. In the first case, the problem is known as a *static* path planning problem whereas, the second case corresponds to a *dynamic* problem (see [1]). From an application point of view, dynamic problems are more interesting than static ones since autonomous vehicles typically have on-board sensors that obtain information about the operating environment dynamically. On the other hand, dynamic problems are more challenging, since they require the use of algorithms that can be implemented *on-line*. In the past, this kind of implementation was restricted by hardware limitations, and therefore the path designers had to rely more or less on *off-line* solutions. With the recent significant advances in computing technology, real time implementation of path planning algorithms has become easier. As a result, new path planning schemes which can be implemented "on the fly" have appeared over the past decade. Path planning schemes using feedback control laws and the D^* algorithm are some of the most significant examples of this era. In this work we deal with the vehicle navigation problem where the vehicle (e.g., ground vehicle, UAV) is operating inside a partially known environment $\mathcal{W}$.

In the next section we introduce the reader to some basic concepts of the path planning problem and present some significant results that have appeared in the literature and are relevant to our problem.

1.2 The Path Planning Problem

The general framework for the path planning problem is as follows: *given a topological space $\mathcal{F} \subset W$ of admissible states $\mathsf{x}_{\mathcal{F}}$, find a path connecting the prescribed initial state $\mathsf{x}_0 \in \mathcal{F}$ with the destination state $\mathsf{x}_f \in \mathcal{F}$ such that the path lies completely inside $\mathcal{F}$*. The construction of such a path can take place using either a continuous representation of $\mathcal{W}$ or a discrete one. Seeing the problem from the geometrical point of view (no kinematic or dynamic constraints) a sufficient condition for a solution path to exist is that the space $\mathcal{F}$ is polygonally (or path) connected, i.e. that for any two configuration inside $\mathcal{F}$ there exist a path which lies completely inside $\mathcal{F}$ connecting these two configurations. In discrete topologies like finite graph structures comprised of nodes, a sufficient condition is that, for any two nodes in $\mathcal{F}$, there exists a sequence of nodes, adjacent to each other, connecting these two nodes.

1.2.1 Planning Inside Continuous Spaces

In applications where dynamic constraints are taken into consideration the most natural approach is to work with continuous spaces. Then the path planning can be treated as a two boundary value problem (the fixed final point and free final time problem) where application of standard control techniques, such as optimal control theory (see [2]), can be employed.

We define the configuration space $\mathcal{W} \subset \mathbb{R}^n$ to be the space that contains all the possible states x of the agent such that

$$\mathcal{W} = \mathcal{F} \oplus \mathcal{O}, \tag{1.1}$$

where $\mathcal{F}$ denote the space of all admissible states $\mathsf{x}_{\mathcal{F}}$ known as the free configuration space and $\mathcal{O}$ the obstacle configuration space which contains all the unfeasible states. Finally, we denote with $\mathcal{U}$ the function space of all admissible inputs u which take values in the space $U \subset \mathbb{R}^m$. Let's assume that the dynamics that

govern the motion of the agent are given in the form

$$\dot{\mathsf{x}}(t) = f(\mathsf{x}(t), u(t)), \quad \mathsf{x}(0) = \mathsf{x}_0 \;\; t \geq 0 \tag{1.2}$$

where we consider the time invariant case for simplicity. We additionally assume that the space $\mathcal{F}$ is open and connected and the function f satisfies a Lipschitz condition for all $\mathsf{x} \in \mathcal{F}$. The condition on $\mathcal{F}$ being a region (open and connected subset of $\mathbb{R}^n$) is equivalent to the polygonally connected requirement (see [3]) introduced earlier whereas the Lipschitz condition on f is needed so that a solution path always exist (but may not be unique, see [4]). Then the path finding problem is the one of finding a control input function $u \in \mathcal{U}_{[0,t_f]}$ that drives the agent from the initial state x_0 to the final state x_f in finite time t_f, while the whole trajectory from x_0 to x_f lies entirely on $\mathcal{F}$.

If an optimal feedback control cannot be derived for our problem then the optimal control solution may not be plausible from the application point of view. This is due to the sensitivity that may be exhibited to variations of the initial and final configurations or to any environment parameters. One popular technique to proceed is the potential function method (see [5, 6]). The potential functions are scalar functions $\phi : \mathcal{F} \mapsto \mathbb{R}$ which attains their global minimum at the final destination state. These sufficiently smooth functions, known also as navigation functions, provide control laws in feedback form which result in smooth collision free trajectories which may given in closed mathematical expressions (analytic solutions). For these reasons navigation function are very efficient for real time implementation. The feedback control law is given by

$$u(t) = -K\nabla\phi(\mathsf{x}(t)) \tag{1.3}$$

where $\nabla\phi(\mathsf{x})$ denotes the gradient vector of the navigation function at state x where $K \in \mathbb{R}$ is a gain matrix. With $\dot{\mathsf{x}}(t) = f(\mathsf{x}(t), -K\nabla\phi(\mathsf{x}(t))) = \tilde{f}(\mathsf{x}(t))$ be the closed loop dynamics, the trajectory $\mathsf{x}(t)$ of the agent at some time t is now given by

$$\mathsf{x}(t) = \mathsf{x}_0 + \int_0^t \tilde{f}(\mathsf{x}(\tau))\mathrm{d}\tau. \tag{1.4}$$

A drawback of this approach is the existence of local minima or even stationary points which may not allow the agent to reach its destination even though a solution path may exist. Under some specific assumptions on the geometry of

the working environment, the structure of obstacles, and the agent's governing equations of motion, results that guarantee that the trajectory of an agent driven by a feedback control law induced by a potential function converges to the desired destination were first presented in [7, 8]. The approach used in the latter method is the construction of a function which attains its minimum value at the destination state. The final state in this case is the unique isolated minimum of $\phi(\mathsf{x})$ which additionally corresponds to an asymptotically stable equilibrium of the closed loop dynamics. Thus, when the agent's configuration is arbitrarily close to the destination state, the agent practically halts there. However, the construction of the potential function in arbitrary geometries of either the configuration or obstacle space without local minima other than the destination state is not an easy task and general results have not been found. Additionally, the task of finding an appropriate potential function becomes more complicated as the complexity of the agent's governing equations is increased.

1.2.2 Planning Inside Discrete Spaces

Due to the existence of many shortest path and network minimization algorithms, planning inside discrete spaces has become very popular in many path planning applications. The discrete space $\mathcal{X}$, which is the equivalent of the continuous configuration $\mathcal{W}$, is defined as the countable collection of all possible states x such that

$$\mathcal{X} = \mathcal{X}_{\text{free}} \oplus \mathcal{X}_{\text{unfeasible}}$$

where $\mathcal{X}_{\text{free}}$ and $\mathcal{X}_{\text{unfeasible}}$ denote the corresponding subcollections of all feasible and unfeasible states $\mathsf{x} \in \mathcal{X}$ respectively. Let now x_k be the state at some time t_k, then the state at the next time step t_{k+1} is given by

$$\mathsf{x}_{k+1} = f(\mathsf{x}_k, u_k) \tag{1.5}$$

where u_k is an action applied at time t_k to the agent with $u_k \in \mathcal{U}$, where $\mathcal{U}$ denotes the countable collection of all available admissible inputs, and f is a state transition function

$$f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$$

which behaves as the 'dynamics' of this discrete system. The problem then becomes one of finding the appropriate sequence of actions u_i which will drive the

agent form an initial state $\mathsf{x}_0 \in \mathcal{X}$ to the final destination $\mathsf{x}_f \in \mathcal{X}$ after a finite number of steps.

A common method to pass from the real environment (continuous space) to the discrete world is by using cell-based approximations of the environment and then employ a graph representation scheme. By transcribing the free space $\mathcal{F}$ on a graph $\mathcal{G}$ the problem reduces to one of finding a sequence of adjacent nodes in the graph $\mathcal{G}$ from the starting node to the destination node. These nodes form a connected sequence of nodes of the graph, provided that such a sequence exists. In case where more than one feasible solutions exist, optimization criteria determine the one that will qualify for the agent's path. The problem is therefore reduced to a network minimization or a graph search problem. Algorithms that find a solution to the graph search problem or prove that the problem has no solution are called *exact* or *complete* algorithms (e.g. Dijkstra's algorithm, A^* algorithm, see [9, 10]). These path planning algorithms will find the solution even when the polygonal connected condition for the entire free space $\mathcal{F}$ is not satisfied, provided of course that a solution exists for the specific choice of the initial and final node. Furthermore, in case where there does not exist a polygonally connected subset of $\mathcal{F}$ that contains the initial and final nodes, then Dijkstra's algorithm and the A^* algorithm will determine that the specific problem is infeasible. Another category of planning algorithms are the heuristic algorithms which generate a solution while aiming at lower computational complexity (see [6, 5]).

All the algorithms we mentioned in the previous paragraph are mostly used to solve static problems. On the other hand, dynamic problems require appropriate modifications to the original static path planning schemes. The basic idea is to deal with the dynamic problem as a sequence of different static problems. Each one of these static problems corresponds to the planning problem inside the environment representation of the specific time step (continuous replanning). A more straightforward approach for the dynamic problem is the D^* algorithm (or Stentz's algorithm) which is presented in [11]. The D^* algorithm is the first path planning algorithm for discrete spaces which deals exclusively with the dynamic problem while having significant advantages over the other available dynamic schemes (e.g. reduced computational complexity, optimality in the global sense). In Stentz's algorithm the operating environment is assumed to be partially known

and therefore the transition costs between nodes of the corresponding graph are originally unknown. The agent's sensors provide information about the agent's immediate environment and this information is processed in a heuristic way. This allows the construction of the true operating map with reduced computational complexity. Subsequently, the algorithm uses a graph search approach, which can be characterized as a dynamic extension of Dijkstra's algorithm, in order to find the optimal solution in a complete way. Improved versions of the D^* algorithm, namely the delayed D^* and the D^* lite algorithms, are presented in [12, 13] respectively.

An important issue that should be dealt within the discrete approach is the "smoothness" requirement of the generated path, since in realistic situations the vehicle under dynamic constraints has to follow a smooth trajectory. In the literature (see [14]) one can find modified shortest path algorithms which assign an additional penalty cost when the angle between two successive arcs of the path is larger than a threshold angle $\theta_{\max}$. By imposing this "smoothness" constraint, the sequence of nodes that solve the network minimization problem form a smooth enough polygonal line joining x_0 and x_f. Then, one can directly proceed to the trajectory planning problem by determining the linear and angular velocities required to drive the agent to the goal destination. The resultant trajectory has to be close enough, in a pointwise sense, to the polygonal path that the graph search algorithm found. This requirement is a direct consequence of the polygonal connected requirement for the space $\mathcal{F}$ since no other curve except from the polygonal path is guaranteed to lie completely inside $\mathcal{F}$.

The most important difficulty of the graph search approach lies in the dimensionality of the problem. To elucidate this point, let us assume that we implement an on line path planning scheme where all the data of the environment are available at all times. On the other hand, the available computational resources are limited due to hardware limitations. It is obvious that the higher the amount of data processed, the more accurate the solution path will be. Thus, the dimension of $\mathcal{G}$ (i.e., the number of nodes) becomes very large as the fidelity of the approximation of $\mathcal{F}$ and/or $\mathcal{W}$ increases. Furthermore, the increase in the number of the nodes of the graph is very likely to increase the adjacency relations between different nodes. In all cases, the graph search task becomes more demanding and

the problem more computationally complex. Sooner or later, the computational resources on-board the agent reach their limit; hence, an accurate solution cannot be generated as often as desired, and therefore the agent capacity to deal with rapidly changing situations is also limited.

1.3 Thesis Motivation and Objectives of The Proposed Path-Planning Schemes

1.3.1 Introduction

From the previous observations, a dynamic path planning scheme which can handle changes in the vicinity of the agent while requiring a reduced amount of computational resources will be useful in many applications in the field of UAV, UGV navigation and robot motion planning. One straightforward approach to deal with this problem is to design a path planning algorithm based solely on local information of the configuration space. In that case replanning is easier than with global methods; there exist obstacle configurations nonetheless, where such local methods are not guaranteed to find a solution even if such a solution exists. This situation is depicted in Fig. 1.1 where the agent starting from 'A' fails to find the route to the goal destination 'B' when the visibility/exploration horizon of its sensors are not sufficient large. As the exploration horizon is increased, the construction of a collision-free path becomes more likely. The increase of the exploration horizon, however, comes at the expense of an increase of the required on-board computational resources. Therefore, a path planning algorithm based on local information may not be efficient in practice unless the initial and final states are sufficiently close to each other (see [1]). On the other hand, global path-planning algorithms in realistic situations, where the agent's computational resources are limited, are restricted to use coarse representations of the whole environment. Thus, the global information approach fails to take into consideration the high fidelity information in the vicinity of the agent's current position. Subsequently, the real time implementation of a global path-planning scheme becomes problematic since the agent is not appropriately sensitive to crucial changes in its immediate environment.

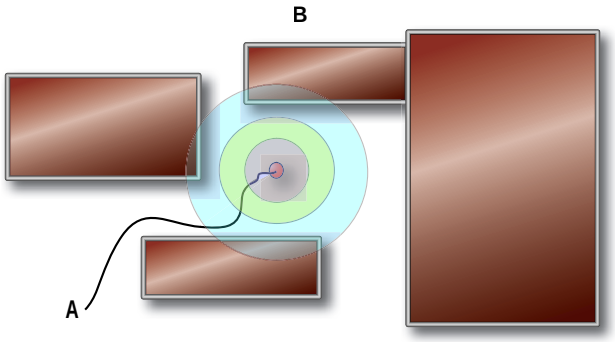


Figure 1.1: A path generated by a short visibility/exploration horizon planning scheme.

1.3.2 Thesis Motivation

Strong motivation for this thesis is the design of a path planning scheme where the operating environment of the vehicle is assumed to be only partially known. This motivation comes from many applications of autonomous vehicle navigation problems, where the assumption of global knowledge of the environment is an overconservative approach. One can think of an unmanned aerial vehicle operating in a hostile environment with the mission to identify an enemy target, where areas of high risk are not known *a priori*. Based on the information gathered by the on-board sensor devices, the UAV should find the route to the destination while avoiding to get close to areas of high risk. It is evident that its ability to react immediately to an obstacle or popup threat is crucial to the success of its mission. In this case, as with many other applications of vehicle navigation, different sensor devices can provide information of the environment in the vicinity of the vehicle. The information obtained by different devices should be weighted appropriately, since the sensors, (cameras, radars, laser scanners, satellite imagery, etc.) have different ranges and resolutions.

1.3.3 Thesis Objectives

In this work, we introduce two hybrid local/global path planning algorithms that use district levels of fidelity (resolution) of the environment at different distances

from the agent's current position. The first planning scheme uses cell decompositions of the agent's operating environment constructed using wavelets and standard quadtree decompositions. The second scheme, finds the solution path using sector cell approximations. This is a more natural approach given the sector like topology of the areas scanned *on-line* by the on-board sensors of the vehicle. In the sequel, when we will use the term the "*path planning scheme*" we will refer to the general idea of multiresolution path planning lying behind both of these algorithms. In cases where we want to distinguish between the two, we do so explicitly.

The goal of the proposed *path planning scheme* is to find a sequence of points in the world $\mathcal{W}$ (operating environment of the vehicle) that the agent should visit in order to reach the final destination x_f (perhaps not accurately known a priori). The initial state x_0 is assumed to be prescribed whereas the final time is a free parameter of the problem. All the points of the sequence should lie in a polygonally connected subspace of the free configuration space $\mathcal{F}$. This requirement is needed so that the polygonal line that connects the points of the sequence lies completely inside $\mathcal{F}$.

Since the resultant path of the *planning scheme* is a finite sequence of ordered points, the next step is to introduce a methodology for generating a smooth trajectory under dynamic constraints. We employ a kinematic ground vehicle model to obtain the equations of motion of the agent, and additionally we introduce a trajectory generating scheme using some basic ideas from input/output linearization theory (IOL) for MIMO systems (see [15]). This scheme produces a smooth curve passing sufficiently close to the visiting points of the path at specified instants of time (time scheduling).

1.3.4 The Multiresolution Hierarchical Approach

The success of a multi-resolution path planning algorithm hinges on its ability to compute the obstacle boundaries well in advance and with sufficient accuracy, by keeping a balance between an overconservative approximation of the environment and the computation of a collision-free path.

In this work we use wavelets to obtain multiresolution approximations of the configuration space at different distances from the vehicle. The computational

complexity of the wavelet transform is of order $O(n)$ where n is the input data [16] while even than the Fast Fourier Transform has complexity of order $O(n \log_2 n)$. Therefore the wavelet transform provides a very fast decomposition of the environment at different levels of resolution. From the output of the wavelet transform we construct cell-based approximations (rectangular or sector-like) of the whole environment $\mathcal{W}$ having high/fine resolution close to the current position of the vehicle and low/coarse resolution far away. The number of resolution levels, their scale, and range can all be readily adapted at each time step to yield graph representations that are commensurable to the available on-board computational resources.

We employ the hierarchical path planning principle to find the optimal path on a topological graph $\mathcal{G}$ induced by the previous cell decomposition. Namely, the path may contain mixed nodes at all resolution levels except the finer resolution level, where it is assumed that nodes can be confidently resolved as either free or occupied. Mixed nodes, on the other hand are not known with certainty whether they belong to the free or the obstacle space. Hierarchical path planning is known to be more flexible than methods that search only through free nodes [17]. In hierarchical path planning the mixed nodes are subsequently resolved to free or occupied nodes, as the agent gets closer to the obstacles and more information about their shape and location becomes available.

1.3.5 Literature Review

Several multi-resolution or hierarchical algorithms have been proposed in the literature for path planning [18, 19] The majority of those use some form of quadtree decomposition of the environment. One drawback of quadtree-based decompositions is that a finer resolution is used close to the boundaries of all obstacles, regardless of their distance from the agent. This tends to waste computational resources. One of the central references in the context of quadtree-based cell decompositions is perhaps [19], where the authors present a hierarchical path planning scheme based on a multistage quadtree decomposition. Both free and mixed nodes are included in the search, which is conducted using the A^* algorithm. A path to the target is first computed using a coarse grid and subsequently refined using information from higher resolution levels, uniformly along the path. Even

though this technique is efficient and easy to implement, it fails to take full advantage of the local information around the agent. Wavelets for multi-resolution decomposition of the environment have also been used in [18]. The approach in [18] combines a more efficient model for the local behavior of the approximation, with improved computational characteristics, compared to the one proposed in [19]. The main emphasis in [19] is to construct a smooth path. This is easily achieved by the information provided by the detail coefficients in the wavelet expansion. The smoothness requirement is then embedded in the transition cost of the agent. The reference most closely related to our approach is [20]. Therein, the author also uses the idea of coarse/fine grid at close/far distances from the current location of the agent in order to avoid the demerits of uniform grids or standard quadtrees. Nonetheless, no connection with wavelets is attempted. In addition, the multiresolution scheme in [20] requires a rather careful handling of the cell connectivity at the boundaries between two different resolution levels. This is handled automatically in our approach.

1.4 Thesis Overview

In this section we briefly describe each chapter in this thesis by and we point out interrelationships between these chapters.

Chapter 2: Technical background

We discuss some basic notions from topology, graph theory and wavelet approximation theory and we present Dijkstra's algorithm and the elementary theory of the quadtree decomposition algorithm. These topics are important for a deeper understanding of the subsequent chapters. A reader familiar with these concepts can skip this chapter in a first reading.

Chapter 3: Multiresolution Path planning Using Rectangular Cell Decompositions

We introduce the first path planning scheme of this thesis. The proposed algorithm constructs the shortest path based on rectangular cell decompositions of the world space $\mathcal{W}$ induced by a *Haar* wavelet multiresolution approximation scheme of a risk measure function defined over $\mathcal{W}$.

Chapter 4: Multiresolution Path Planning Using Sector Decomposi-

tions

We present the second path planning scheme of this thesis, which is a natural extension of the algorithm presented in Chapter 3. Now the planning takes place in sector-cell decompositions of the world $\mathcal{W}$. This approach is compatible with the specific form of data obtained by on-board sensors of an autonomous vehicle.

Chapter 5: Time Scheduling and Smooth Trajectory Generation

We discuss ways to generate a smooth trajectory passing sufficiently close to the visiting points given by the *path planning scheme* presented in the previous two chapters. We wish this closeness requirement to be satisfied at specific time instants. We first introduce a kinematic model to describe the equations of motion of the agent, and we subsequently specify an appropriate speed profile to accomplish the trajectory generation and time scheduling tasks. Ideas from input/output linearization theory for MIMO systems are employed to achieve this objective.

Chapter 6: Simulation Results

We present simulation results to test the efficiency of the two proposed planning algorithms and the trajectory generating and time scheduling scheme.

Chapter 7: Conclusions and Future Work

We discuss the conclusions from our approach to the path planning problem. Ideas and possible extensions are also briefly discussed.

Chapter 2

Technical background

2.1 Introduction

In this chapter we present some basic notions that we will use extensively in this work. First, we present the basic theory of cell decomposition schemes, which are typically used in robot motion planning application [5, 6] and we describe how we incorporate the cell decomposition approach in our problem. Then we introduce the elementary theory of the wavelet transform. In this work we employ wavelet schemes to obtain cell decompositions of the working environment with special localized attributes. Furthermore, we present the basic ideas behind the Dijkstra's algorithm which we use to solve the shortest path problem. For a more detailed presentation of the shortest path problem the reader is encouraged to consult the suggested references relevant to this topic [21, 9].

2.2 Cell Decomposition

An m-cell decomposition $\mathcal{C}_d$ of $\mathcal{W}$ is a finite collection of m cells

$$\mathcal{C}_d = \{c_i \in \mathcal{W} : \ i = 1, \ldots, m\} \tag{2.1}$$

with the following properties:

1. $\mathcal{W} = \bigcup_{i=1}^{m} c_i$
2. $\text{int}(c_i) \bigcap \text{int}(c_j) = \varnothing$

A cell decomposition algorithm generates a collection of cells by creating sequences of nested cells. One popular algorithm to accomplish this decomposition is the quadtree decomposition algorithm. The goal of this algorithm is the construction of a collection of square cells which contains only empty or full cells. In order to present the way the algorithm constructs these sequences, let us suppose that initially we have only one mixed cell c_1 that encloses the world $\mathcal{W}$ with ℓ_1 be the length of each of its sides. Then by bisecting each of the sides of the cell c_1 we take four new cells c_i, with $i = 1, \ldots, 4$ with corresponding side length $\ell_i = \ell_1/2$. The cell c_1 is the parent node and c_i are the children nodes of the corresponding expansion tree of depth $k = 2$. Since the aim of the cell decomposition algorithm is to generate only empty or full cells the algorithm will continue to subdivide each mixed cell creating four new cells with sides of length $\ell_i = \ell_1/2^{k-1}$, where k the depth of the corresponding expansion tree. The algorithm will terminate either when there no mixed cells or the resulting expansion tree reaches a predetermined depth.

Given two cell decompositions $\mathcal{C}_d$ and $\mathcal{C}'_d$ of $\mathcal{W}$ we say that $\mathcal{C}'_d$ is a *finer, or higher resolution* decomposition of $\mathcal{W}$ than $\mathcal{C}_d$ if and only if for every cell $c_i \in \mathcal{C}_d$ there exists an integer $p_i > 1$ such that $c_i = \bigcup_{l=1}^{p_i} c'_l$ where $c'_l \in \mathcal{C}'_d$.

We may define the following three categories of cells:

1. empty cells, when $c_i \cap \mathcal{O} = \varnothing$
2. mixed cells, when $c_i \cap \mathcal{O} \neq \varnothing$
3. full cells, when $c_i \subseteq \mathcal{O}$.

We will say that two cells c_i and c_j are adjacent if

$$\partial c_i \cap \partial c_j \neq \varnothing, \quad i \neq j, \tag{2.2}$$

where ∂c_i denotes the boundary of the cell c_i. A cell decomposition of a square environment is presented in Fig. 2.1. The corresponding to this decomposition free, mixed and full cells are given in the Table 2.1.

2.3 Graph Representation

To a cell decomposition $\mathcal{C}_d$ we will associate a directed graph $\mathcal{G} = (V, E)$ with nodes V and edges E, known as the connectivity graph, such that:

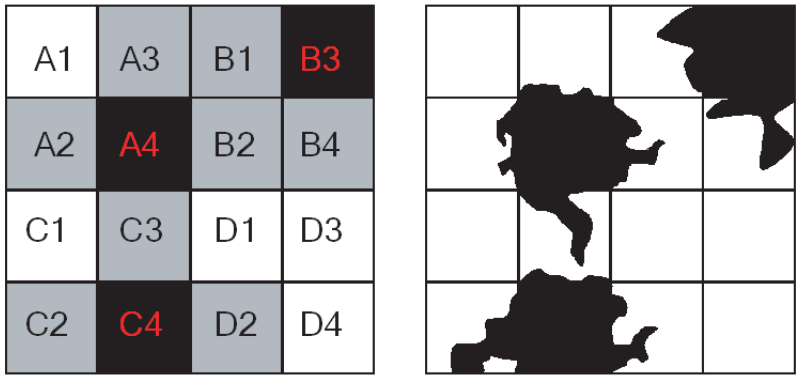


Figure 2.1: Quadtree decomposition scheme. Right figure depicts the world $\mathcal{W}$ where the white areas correspond to $\mathcal{F}$ and black to $\mathcal{OB}$. In the left figure we see the obtained cell decomposition after we apply the quadtree algorithm.

Table 2.1: Cell characterization for the decomposition depicted in Fig. 2.1.

Full Cells	A4, B3, C4
Mixed Cells	A2, A3, B1, B2, B3, B4, C2, C3, D2
Free Cells	A1, C1, D1, D2, D3

1. The nodes of $\mathcal{G}$ correspond to the free and mixed cells of $\mathcal{C}_d$
2. The edges of $\mathcal{G}$ correspond to cells that are adjacent to each other

It is easy to see that $\mathcal{G}$ is a topological graph [5].

In this work we use the 8-connectivity scheme to define adjacency relationships between nodes. This is an immediate result of the equation (2.2). This adjacency scheme is presented in Fig. 2.2. To each ordered pair inside E we associate a cost of transition. In our *path planning scheme* the order of a transition is important for the cost assignment procedure, i.e the connectivity graph is in our case a directed graph.The procedure of cost assignment is described in detail in the subsequent chapters. Finally, the reader interested in a deeper understanding of topics from graph theory should refer to [22, 23, 24].

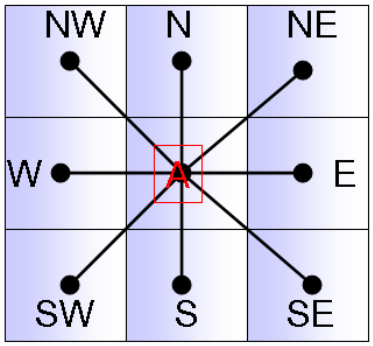


Figure 2.2: The 8-connectivity adjacency scheme.

2.4 The 2D Wavelet Transform

The idea behind the theory of the wavelet transform is to represent a function $f \in \mathcal{L}_2(\mathbb{R})$ as a summation of elementary basis functions $\phi_{J,k}$ and $\psi_{j,k}$ as follows

$$f(x) = \sum_{k\in\mathbb{Z}} a_{J,k}\phi_{J,k}(x) + \sum_{j\geq J}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x), \tag{2.3}$$

where $\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k)$ and $\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$. In the ideal case both $\phi(x)$ (scaling function) and $\psi(x)$ (mother wavelet) have compact support or they decay very fast outside a small interval so they can capture localized features of f. The first summation in (2.3) gives a low resolution, or coarse, approximation of f. The second term in (2.3) gives the difference (details) between the original function and its low resolution approximation. For example, when analyzing a signal at the coarsest level (low resolution) only the general, most salient, features of the signal will be revealed. The index j denotes the resolution level. For each increasing index j, a higher, or finer resolution term is added, which adds more and more details. The expansion (2.3) thus reveals the properties of f at different levels of resolution [25, 26, 27].

This idea can be readily extended to the two-dimensional case by introducing

the following families of functions

$$\Phi_{j,k,\ell}(x,y) = \phi_{j,k}(x)\phi_{j,\ell}(y) \tag{2.4}$$

$$\Psi^1_{j,k,\ell}(x,y) = \phi_{j,k}(x)\psi_{j,\ell}(y) \tag{2.5}$$

$$\Psi^2_{j,k,\ell}(x,y) = \psi_{j,k}(x)\phi_{j,\ell}(y) \tag{2.6}$$

$$\Psi^3_{j,k,\ell}(x,y) = \psi_{j,k}(x)\psi_{j,\ell}(y) \tag{2.7}$$

Given a function $f \in \mathcal{L}_2(\mathbb{R}^2)$ we can then write

$$\begin{aligned} f(x,y) = &\sum_{k,\ell\in\mathbb{Z}} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) \\ &+ \sum_{i=1}^{3}\sum_{j\geq J}\sum_{k,\ell\in\mathbb{Z}} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y) \end{aligned} \tag{2.8}$$

where, for the case of orthonormal wavelets the approximation coefficients are given by

$$a_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\phi_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{2.9}$$

and the detail coefficients by

$$d^i_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Psi^i_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{2.10}$$

In the more general case, biorthogonal wavelets projections on the space spanned by the dual wavelets and dual scaling functions should be used in (2.9) and (2.10). The key property of wavelets used in this work is the fact that the expansion (2.8) induces the following decomposition of $\mathcal{L}_2(\mathbb{R}^2)$

$$\mathcal{L}^2(\mathbb{R}^2) = \mathcal{V}_J \oplus \mathcal{W}_J^{detail} \oplus \mathcal{W}_{J+1}^{detail} \oplus \cdots \tag{2.11}$$

where $\mathcal{V}_J = \overline{\text{span}}\{\phi_{J,k,\ell}\}_{k,\ell\in\mathbb{Z}}$ and similarly $\mathcal{W}_j^{detail} = \overline{\text{span}}\{\psi^1_{j,k,\ell}, \psi^2_{j,k,\ell}, \psi^3_{j,k,\ell}\}_{k,\ell\in\mathbb{Z}}$ for $j \geq J$.

By using the Haar family of wavelets, each scaling function $\phi_{j,k}(x)$ and wavelet function $\psi_{j,k}(x)$ in the Haar system is supported on the dyadic interval $I_{j,k} \stackrel{\triangle}{=} [k/2^j, (k+1)/2^j]$ of length $1/2^j$ and does not vanish in this interval [25, 28]. Subsequently, and via the tensor product in (2.4), we may associate the functions $\Phi_{j,k,\ell}$ and $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ in the 2D case with the square cell $c^j_{k,\ell} \stackrel{\triangle}{=} I_{j,k} \times I_{j,\ell}$.

2.5 Description of Dijkstra's Algorithm

A popular algorithm to find the shortest path between two nodes u and v of a graph $\mathcal{G}$ is the Dijkstra's algorithm. Dijkstra's algorithm is considered as one of the most efficient algorithms when a path between two nodes of a graph is needed (see [1]). Even though it is based on a greedy strategy the Dijkstra's algorithm always finds the optimal solution, provided that such a solution exists. Additionally, depending on the sparsity of the adjacency matrix and the particular implementation it is computationally more appealing than other standard shortest path algorithms, like the Bellman-Ford algorithm, see [9].

Below, we shall briefly present the basic principles of Dijkstra's algorithm. Given a graph $\mathcal{G}$, we associate to a transition from the node v_{i-1} to the node v_i of this graph a nonnegative cost $\mathcal{J}(v_{i-1}, v_i)$. The exact cost assignment procedure shall be described in the subsequent chapters where we introduce the multiresolution *path planning scheme*. The goal of the algorithm is to find the sequence of $1 + m_d$ nodes $\mathcal{P} = (v_0 = s, v_1, \ldots, v_{m_d} = d)$, that the agent has to visit, starting from the initial node s until reaches the destination node d, while at the same the total cost

$$\mathcal{H}(d) = \sum_{j=1}^{m_d} \mathcal{J}(v_{j-1}, v_j) \tag{2.12}$$

is minimized whenever $v_0 = s$. We call $\mathcal{P}$ the shortest path. It is clear that the solution path $\mathcal{P}$ inside the graph $\mathcal{G}$ is a function of the initial and final nodes. When part of the optimal path $\mathcal{P}$ is known we define the optimal weight $\hat{\mathcal{H}}(u_k, u_{k+n})$ to be the sum

$$\hat{\mathcal{H}}(v_k, v_{k+n}) = \sum_{j=k+1}^{n+k} \mathcal{J}(v_{j-1}, v_j), \tag{2.13}$$

where all the nodes appearing in the transition costs of the sum are elements of the optimal part of the path. When v_k is the starting node s then we denote the optimal weight between s and the node $v_{k+n} = u$ as $\hat{\mathcal{H}}(u)$. Dijkstra's algorithm finds $\hat{\mathcal{H}}(u)$ for all $u \in V$ until the value of $\hat{\mathcal{H}}(d)$ is known. This is accomplished with a help of a list S which contains all the nodes u for which $\hat{\mathcal{H}}(u)$ is known. At each iteration the algorithm picks one element w from V which is not in S and is closest to the finite set S. The node w is then an element of the list S. The way that the algorithm assigns the optimal value to the last node w which

was inserted in the list S is based on the *principle of relaxation*. To see how the *principle of relaxation* works we let u, w be two adjacent nodes and assume that the set $S_x(u,w) \stackrel{\triangle}{=} \{x \in V : x \text{ is adjacent to both } u, w\}$ is nonempty. Let $x \in S_x(u,w)$, then the principle is defined by the following inequality

$$\mathcal{H}(u,v) \leq \mathcal{H}(u,x) + \mathcal{H}(x,v). \tag{2.14}$$

In the case where $u \in S$ the *principle of relaxation* states that $\mathcal{H}(u,w) = \hat{\mathcal{H}}(u,w)$ provided that the inequality (2.14) holds for every node $x \in S_x$. Otherwise $\hat{\mathcal{H}}(u,v) = \min_{x \in S_x}\{\mathcal{H}(u,x) + \mathcal{H}(x,v)\}$. For details the reader can refer to [9].

Chapter 3

Multiresolution Path planning Using Rectangular Cell Decompositions

3.1 Introduction

In this chapter we introduce an innovative approach to the path planning problem using ideas from wavelet theory. With this approach the agent is capable of reacting immediately to situations where the collision avoidance requirement is violated (i.e an obstacle is revealed to be in the agent's vicinity) or a threat suddenly appears (popup threat) while approaching the goal destination. The proposed algorithm assumes that far away obstacles or threats should not have a large effect on the vehicle's *immediate* motion, since either these regions will never be visited by the agent or more accurate and reliable information about them will become available when the agent gets closer to them while approaching the final destination. So high resolution information is needed only for the area in the vicinity of the agent's current position. This allows one to construct a graph with fewer nodes than the case where the whole environment is represented in a uniform fashion. One should note that the approach of using a different resolution to approximate the working environment is also based in practical issues since in many typical navigation problem a plethora of sensory devices used (e.g., cameras, radars, laser scanners, satellite imagery) have different ranges and resolutions. A

computationally efficient path planning algorithm should be able to combine the information provided by all these sensors in such a way that the computational resources are used on that part of the path (spatial and temporal) that needs them most. Additionally, information about far away obstacles is taken into some consideration providing the planning scheme with a mechanism which "pulls" the agent to the final destination (long-term strategy).

The resultant solution-path of this algorithm is the one that minimizes the total length of the path with respect to a metric that includes actual path length along with a risk-induced metric. The risk-induced metric depends on the available environment representation and the way is defined is presented in detail in Section 3.3.

3.2 Wavelet decomposition of the risk measure

Without loss of generality, we take $\mathcal{W} = [0,1] \times [0,1]$, which is described using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from the previous discussion that the Haar wavelet decomposition of a function f defined over $\mathcal{W}$ at resolution level $J \geq J_{\min}$

$$\begin{aligned} f(x,y) = & \sum_{k,\ell=0}^{2^{J_{\min}}-1} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(x,y) \\ & + \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J-1} \sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y) \end{aligned} \tag{3.1}$$

induces a cell decomposition of $\mathcal{W}$ of square cells of size $1/2^J \times 1/2^J$.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto [0,1]$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$. For instance, one may choose

$$\mathsf{rm}(\mathsf{x}) = \begin{cases} (d_{\max} - \min_{\mathsf{y} \in \mathcal{O}} \|\mathsf{x} - \mathsf{y}\|_\infty)/d_{\max}, & \text{if } \mathsf{x} \in \mathcal{F}, \\ 1, & \text{if } \mathsf{x} \in \mathcal{O}, \end{cases} \tag{3.2}$$

where $d_{\max} \stackrel{\triangle}{=} \max_{\mathsf{x} \in \mathcal{F}} \min_{\mathsf{y} \in \mathcal{O}} \|\mathsf{x} - \mathsf{y}\|_\infty$. Alternatively, one may think of rm as the probability that $(x,y) \in \mathcal{O}$.

Let us also assume that we have m distinct risk measure levels, say $M_1 < M_2 < \cdots < M_m$ where $M_i \in [0,1],\ i = 1, \ldots, m$.

We will use the $\|\cdot\|_\infty$ norm to measure distances in $\mathcal{W}$. Consequently, all points within range r from the current location of the agent are given by

$$\mathcal{N}(\mathsf{x}, r) \stackrel{\triangle}{=} \{\mathsf{y} \in \mathcal{W} : \|\mathsf{x} - \mathsf{y}\|_\infty \leq r\}. \tag{3.3}$$

Suppose now that we are given the desired levels of resolution of $\mathcal{W}$ as $J_{\min} \leq j \leq J_{\max}$ where $J_{\min}, J_{\max} \in \{1, \ldots, N\}$, with corresponding ranges $r(j)$ from the agent's current location. By this we mean that we wish all points $\mathsf{y} \in \mathcal{N}(\mathsf{x}, r(J_{\max}))$ to be described by resolution $J_{\max}$, all points $\mathsf{y} \in \mathcal{N}(\mathsf{x}, r(j-1)) \backslash \mathcal{N}(\mathsf{x}, r(j))$ to be described by resolution j, where $J_{\min} < j \leq J_{\max}$, and all points $\mathsf{y} \notin \mathcal{N}(\mathsf{x}, r(J_{\min}+1))$ to be described by resolution $J_{\min}$. Since we require finer resolution closer to the agent we assume, of course, that $r(j-1) > r(j)$. The situation is depicted in Fig. 3.1.

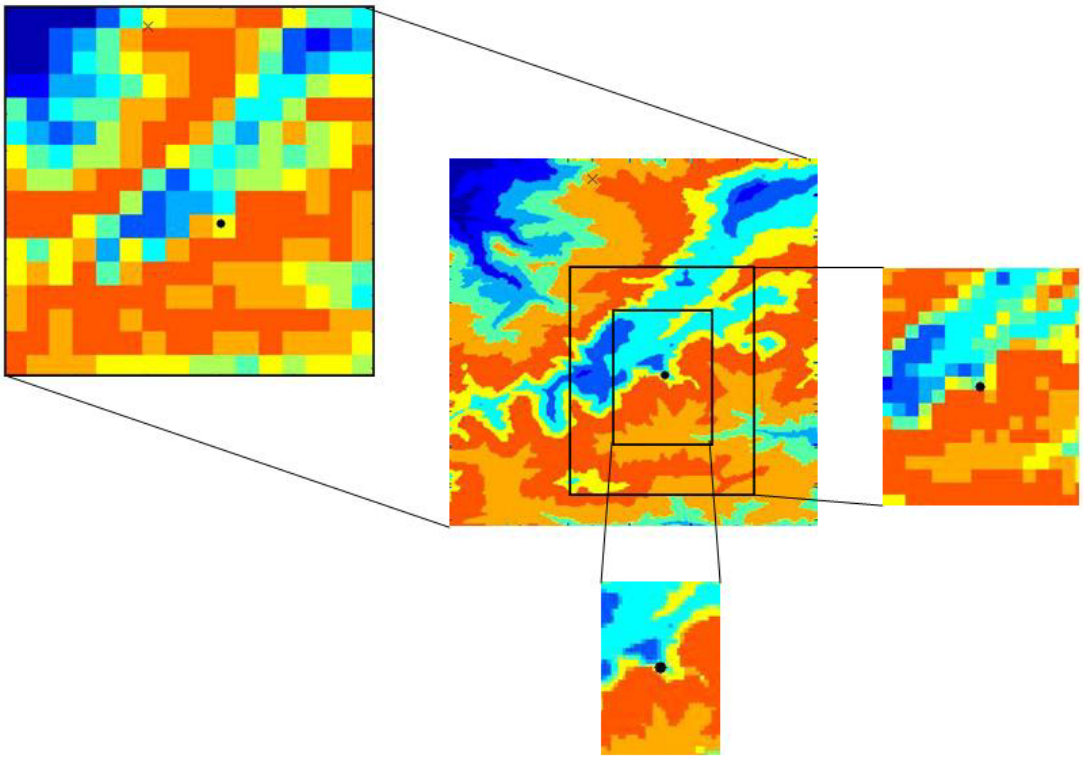


Figure 3.1: Multiresolution representation of the environment according to the distance from the current location of the agent.

The choice of $J_{\max}$ is dictated by the requirement that at this level all cells can be resolved into either free or occupied cells. The choice of $J_{\min}$ as well as the

values of $r(j)$ are typically dictated by the on-board computational resources.

We obtain the distinct resolution levels at the given required distances from the current location of the agent by applying the Haar wavelet transform to rm. The use of Haar wavelets is mainly dictated by the choice of the norm in (4.5). To this end, let $\mathcal{I}(j) \triangleq \{0, 1, \ldots, 2^j - 1\}$ and let

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,k} \cap [x_0 - r(j), x_0 + r(j)] \neq \varnothing\},$$
$$\mathcal{L}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,\ell} \cap [y_0 - r(j), y_0 + r(j)] \neq \varnothing\}.$$

The wavelet decomposition of rm, given by

$$\begin{aligned}\mathsf{rm}(x, y) = &\sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(x, y) \\ &+ \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k \in \mathcal{K}(j) \\ \ell \in \mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x, y)\end{aligned} \tag{3.4}$$

induces, via a slight abuse of notation, the following cell decomposition on $\mathcal{W}$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}}. \tag{3.5}$$

where, ΔC_d^j is a union of cells $c^j_{k,\ell}$ of dimension $1/2^j \times 1/2^j$.

3.3 Cost Assignment

Each cell $c^j_{k,\ell}$ in the cell decomposition has a value $\mathsf{val}(c^j_{k,\ell}) \in \{M_1, \ldots, M_N\}$, which for the case of Haar wavelets is the weighted average of the risk measure function over the cell. Following the hierarchical approach, we divide the cells in three categories: free cells if $\mathsf{val}(c^j_{k,\ell}) < m_1$, full cells if $\mathsf{val}(c^j_{k,\ell}) > m_2$, and mixed cells if $m_1 \leq \mathsf{val}(c^j_{k,\ell}) \leq m_2$, where $m_1, m_2 \in \{M_1, \ldots, M_N\}$ are given.

To the cell decomposition $\mathcal{G}$ having as nodes $V(\mathcal{G})$ all the cells with $\mathsf{val}(c^j_{k,\ell}) \leq m_2$. The edges $E(\mathcal{G})$ of $\mathcal{G}$ correspond to the adjacency relationships of $V(\mathcal{G})$, as usual. Clearly, there is an one-to-one correspondence between the elements of $V(\mathcal{G})$ and the free and mixed cells of $\mathcal{C}_d$. We write $v \sim c^j_{k,\ell}$ to denote this correspondence. Moreover, since $\mathcal{G}$ is a topological graph we may associate each node $v \in V(\mathcal{G})$ with any point $\mathsf{x} \in c^j_{k,\ell}$. Without loss of generality we choose the

center of the cell. Let $\mathsf{cell}_{\mathcal{G}}(v)$ denote the center of the corresponding cell in this case. Finally, if $\mathsf{x} \in c^j_{k,\ell}$ we will write $v = \mathsf{node}_{\mathcal{G}}(\mathsf{x})$ where $v \sim c^j_{k,\ell}$.

To each edge $(u, v) \in E(\mathcal{G})$ we assign a cost $\mathcal{J}(u, v)$, which is the cost of transitioning from node u to node v. We may use the transition cost as follows

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)). \tag{3.6}$$

That is, the cost of transitioning from node u to the adjacent node v depends only on the risk measure of the final node, v and is independent of the starting node u. This situation is depicted in Fig. 3.2. Note that, in general, $\mathcal{J}(u, v) \neq \mathcal{J}(v, u)$.

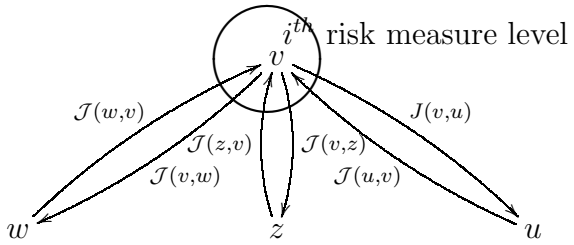


Figure 3.2: Cost assignment for each node u of the directed graph $\mathcal{G}$.

Another alternative would be to choose the transition cost so as to also penalize the (euclidean) distance between $\mathsf{cell}_{\mathcal{G}}(u)$ and $\mathsf{cell}_{\mathcal{G}}(v)$. In this case the cost becomes

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \alpha \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2. \tag{3.7}$$

where $\alpha \geq 0$ is a weight constant. The larger the α the more emphasis we place on a shorter path.

Suppose now that we are given a path of q consecutive, adjacent nodes in $\mathcal{G}$ as follows $\mathcal{P} = (v_0, v_1, \ldots, v_q)$. We can then assign a cost to each node in the path $\mathcal{P}$, induced by the two-node transitioning cost, iteratively, via

$$\mathcal{H}(v_i) = \mathcal{H}(v_{i-1}) + \mathcal{J}(v_{i-1}, v_i), \quad i = 1, \ldots, q. \tag{3.8}$$

The value of $\mathcal{H}(v_k)$ represents the (accumulated) cost of the path from v_0 to v_k ($k \leq q$). The shortest path problem is then to find a path that minimizes the accumulated cost from the initial to the destination node, or determine that such a path does not exist.

3.4 Multiresolution Path planning

The proposed multiresolution path planning algorithm proceeds as follows. Starting from $\mathsf{x}(t_0) = \mathsf{x}_0$ at time $t = t_0$, we construct using the approach of Chapter 2, a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}$. Let the corresponding graph be $\mathcal{G}(t_0)$ and let $v_1^0 \in \mathcal{G}(t_0)$ and $v_f^0 \in \mathcal{G}(t_0)$ be the initial and final nodes, respectively such that $v_1^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_0)$ and $v_f^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_f)$. Using Dijkstra's algorithm (or any other similar algorithm) we find a path $\mathcal{P}(t_0)$ in $\mathcal{G}(t_0)$ of free and mixed nodes from v_1^0 to v_f^0 assuming that such a path exists. Note that areas far away from the agent's current position are likely to be represented by cells of relative high size.Let the path $\mathcal{P}(t_0)$ be given by the ordered sequence of l_0 nodes as follows

$$\mathcal{P}(t_0) = (v_1^0, v_2^0, \cdots, v_{l_0-1}^0, v_{l_0}^0 = v_f^0).$$

It is assumed that v_2^0 is free owing to the high resolution decomposition of $\mathcal{W}$ close to x_0. The agent subsequently moves from v_1^0 to v_2^0. Let now t_1 be the time the agent is at the location $\mathsf{x}(t_1) = \mathsf{cell}_{\mathcal{G}(t_0)}(v_2^0)$ and let $\mathcal{C}_d(t_1)$ be the multiresolution cell decomposition of $\mathcal{W}$ around $\mathsf{x}(t_1)$ with corresponding topological graph $\mathcal{G}(t_1)$. Applying again Dijkstra's algorithm we find a (perhaps new) path in $\mathcal{G}(t_1)$ from $v_1^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1))$ to $v_f^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}_f)$ if such a path exists. Let $\mathcal{P}(t_1)$ be given by the ordered sequence of l_1 nodes as follows

$$\mathcal{P}(t_0) = (v_1^1, v_2^1, \cdots, v_{l_1-1}^1, v_{l_1}^1 = v_f^1).$$

The agent subsequently moves to $\mathsf{x}(t_2) = \mathsf{cell}_{\mathcal{G}(t_1)}(v_2^1)$ at time t_2.

In general, assume the agent is at location $\mathsf{x}(t_i)$ at time t_i. We construct a multiresolution decomposition $\mathcal{C}_d(t_i)$ of $\mathcal{W}$ around $\mathsf{x}(t_i)$ with corresponding graph $\mathcal{G}(t_i)$. Dijkstra's algorithm yields a path $\mathcal{P}(t_i)$ in $\mathcal{G}(t_i)$ of mixed and free noes of length l_i,

$$\mathcal{P}(t_i) = (v_1^i, v_2^i, \cdots, v_{l_i-1}^i, v_{l_i}^i = v_f^i),$$

where $v_1^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}(t_i))$ and $v_f^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}_f)$ if such a path exists. The process is continued until some time t_f when $\|\mathsf{x}(t_f) - \mathsf{x}_f\| < 1/2^{J_{\max}}$, at which time the algorithm terminates. At the last step the agent moves from $\mathsf{x}(t_f)$ to x_f.

Note that the actual path $\mathsf{x}(t_0), \mathsf{x}(t_1), \ldots, \mathsf{x}(t_f)$ followed by the agent is given by the sequence of nodes $\mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}(t_0)), \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1)), \ldots, \mathsf{node}_{\mathcal{G}(t_f)}(\mathsf{x}(t_f))$. Since

the connectivity graph $\mathcal{G}(t)$ changes at each time step, it is therefore possible that the same state x may be visited twice since it may correspond to nodes of two distinct graphs. That is, it is possible that

$$\mathsf{cell}_{\mathcal{G}(t_i)}(v_k^i) = \mathsf{cell}_{\mathcal{G}(t_j)}(v_m^j), \quad i \neq j. \tag{3.9}$$

This will cause the agent to repeat the previous (optimal) decision ending up in a continuous loop. In order to avoid such pathological situations, we maintain a list $\mathsf{L}_{\mathsf{Visited}} = \{\mathsf{x}(t_0), \mathsf{x}(t_1), \ldots, \mathsf{x}(t_i)\}$ of all visited states up to the current time step t_i. At the next time step t_{i+1} we remove from $V(\mathcal{G}(t_{i+1}))$ all nodes v such that

$$\mathsf{cell}_{\mathcal{G}(t_{i+1})}(v) \in \mathsf{L}_{\mathsf{Visited}}. \tag{3.10}$$

A pseudo-code implementation of the above algorithm is given in Fig. 3.3.

```
BEGIN PATH PLANNING ALGORITHM
{
    i = 0;
    x_i ⟵ x_0;
    {L_Visited} = {∅};
    (while ‖x_f − x_i‖ > ε)
    {
      compute rm(x, i) for all x ∈ W;
      construct C_d(i);
      construct G(i) = (E(i), V(i));
      (if L_Visited is nonempty)
        for v ∈ V(i)
          x_v(i) = cell_{G(i)}(v);
          if x_v(i) ∈ L_Visited
            extract v from V(i);
            for all u adjacent to v
              remove (v, u) from E(i);
          end if;
      end if;
      v_1^i ⟵ node_{G(i)}(x_0);
      v_f^i ⟵ node_{G(i)}(x_f);
      P(i) ⟵ Dijkstra(v_1^i, v_f^i, V(i), E(i));
      if P(i) = {∅}
        report FAILURE;
        break;
      x_i(1) = cell_{G(i)}(v_1^i);
      x_i(2) = cell_{G(i)}(v_2^i);
      {L_Visited} ⟵ {L_Visited} ⨁ {x_i(1), x_i(2)};
      x_{i+1} ⟵ x_i(2);
      i ⟵ (i + 1);
      x_0 ⟵ x_i(2);
    }
}
END PATH PLANNING ALGORITHM
```

Figure 3.3: Pseudo-code implementation of proposed multiresolution path planning scheme.

Chapter 4

Multiresolution Path Planning Using Sector Decompositions

4.1 Introduction

In path-planning problems information about the environment is obtained using either on-board or off-board sensors. Some of this information is provided off-line and some is gathered on-line. Furthermore, most typical sensor devices provide sector-like representations of the environment (see Fig. 4.1). This type of information is not in the most efficient form for the majority of planning algorithms, which employ rectangle or square cell approximations, typically using quadtrees [19, 17, 29]. Such approximations are not compatible to the sector based representations obtained by most sensor devices.

In this chapter we present a planning algorithm as an extension of the results of the previous chapter. In the proposed path planning scheme we employ a conformal mapping to devise a hybrid local/global path planning algorithm using sector cell decompositions instead of decompositions that employ only rectangular or square cells. Sector cells are compatible to the on-board sensors and thus process the data more efficiently, in a manner that does not contradict its original sector-based form. We provide approximations with special localized attributes by combining efficiently data from sensors of different resolutions and ranges. Furthermore, in the algorithm of Chapter 3 the whole environment was assumed to be known *á priori* and the wavelet approximation scheme allowed us to plan the path

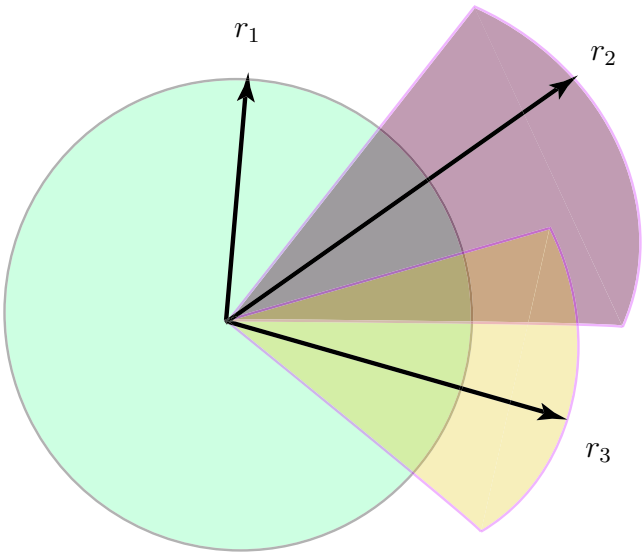


Figure 4.1: Sensors have different ranges, fields of view and resolution. Ideally, the algorithm that processes this data should conform to this topology.

using only a small fraction of the available information. This results in a reduced number of computations that can be handled by the available computational resources on-board the vehicle. Contrary to the planning algorithm of Chapter 3, the proposed methodology in this chapter employs on-line data of the agent's immediate environment as it is obtained by the on-board sensors. This approach is the most natural way to deal with the navigation problem of an autonomous vehicle operating in an unexplored environment, for which no prior knowledge is available.

In the next section we introduce a methodology for obtaining sector cell decompositions given rectangular cell decompositions using conformal mapping. In Section 4.3 we describe the way we represent hierarchically the new world space $\mathcal{W}$ in the new coordinate system induced by the conformal mapping. The new multiresolution decomposition scheme of the risk measure in the new coordinate system is described in Section 4.4. Finally, in Section 4.5 we present in detail the new sector-based multiresolution path planning algorithm.

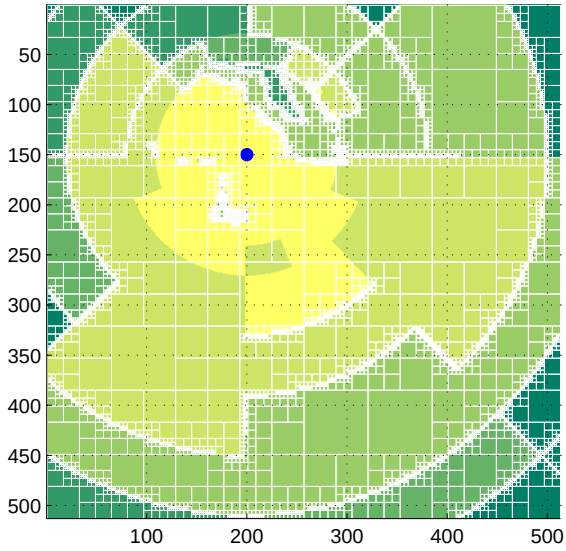


Figure 4.2: A cell decomposition based on the available sector approximation of the environment obtained by the on-board sensor devices of the agent (denoted with the blue dot). In order to resolve the geometry of the arc-boundary of each sector the standard quadtree algorithm generates a large number of cells at close to the boundaries of these arcs.

4.2 From Rectangular to Sector Cell Decompositions

It is assumed that the available information of the surrounding area of the agent is given by the superposition of circular or conical sectors obtained by different sensor devices as depicted in Fig. 4.1. This information about $\mathcal{W}$ needs to be processed by the path planner to compute a collision free path. In order to do so, the path-planning algorithm typically computes a cell decomposition of the environment. As can be easily observed by Fig. 4.2, if rectangular cells are used (as with any quadtree-based approach) the algorithm may waste computational resources by subdividing the cells in order to resolve the sector boundaries.

A simple way to overcome this difficulty is to employ a conformal mapping to map the sector cells to rectangular cells in a new coordinate system. The latter approach is proposed in this chapter. The motivation for this idea is simple. Let

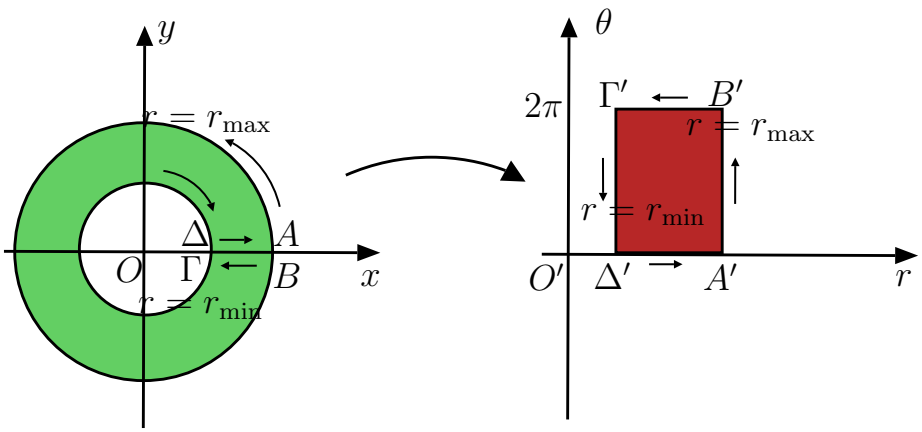


Figure 4.3: A cut annulus in the (x, y) plane is mapped to a rectangle in the (r, θ) plane using a polar (conformal) mapping.

us assume that $\mathcal{R}$ is a sector domain in the (x, y) plane specified by the radii $r_{\min}$ and $r_{\max}$ and the angles $\theta_{\min}$ and $\theta_{\max}$. Mapping this domain to the (r, θ) plane using the polar transformation one obtains a rectangular domain $\mathcal{R}'$ defined by the same radii and angles. If the angle varies from $\theta_{\min} = 0$ up to $\theta_{\max} = 2\pi$ the whole annulus cut defined by the radii $r_{\min}$ and $r_{\max}$ is mapped to a rectangular cell in the (r, θ) plane as shown in Fig. 4.3.

More precisely, recall that the polar coordinates r, θ are related to x and y via the equations

$$x = r\cos\theta, \qquad y = r\sin\theta, \tag{4.1}$$

where the *Jacobian* of the transformation is given by

$$J = \frac{\partial(x, y)}{\partial(r, \theta)} = r,$$

and thus $J > 0$ provided that $r > 0$. Thus, the inverse transformation $(x, y) \mapsto (r, \theta)$ exists for $r > 0$.

The closed curve defined by the union of the circles $C_1 = \{(x, y) \in \mathbb{R}^2,\ x^2+y^2 = r_{\min}^2\}$, $C_2 = \{(x, y) \in \mathbb{R}^2,\ x^2 + y^2 = r_{\max}^2\}$ and the line segment $L = \{(x, y) \in \mathbb{R},\ r_{\min} \le x \le r_{\max}\}$ (traversed twice), maps to a rectangle $r = r_{\min}, r = r_{\max}, \theta = 0, \theta = 2\pi$ in the (r, θ) plane. Alternatively, the area inside a rectangle defined by $r = r_{\min}, r = r_{\max}, \theta = \theta_{\min}, \theta = \theta_{\max}$ where $0 \le \theta_{\min} \le\ \theta_{\max} \le 2\pi$ is mapped via the inverse transformation to the intersection of two sectors defined by $r_{\min}, \theta_{\min}$ and $r_{\max}, \theta_{\max}$ respectively in the (x, y) plane.

Thus, given a sector cell decomposition in the (x, y) plane, a rectangle cell decomposition can be obtained in the (r, θ) plane via the polar coordinate transformation (4.1). Conversely, if we have a multiresolution approximation of the rectangular domain in the (r, θ) plane, then by applying the inverse mapping, the rectangular area can be approximated by a collection of cells forming a sector domain in the (x, y) plane, as seen in Fig. 4.4.

4.3 World Space in the New Coordinate System

Let f be a function $f : \mathfrak{Domain}(f) = \mathcal{W} \mapsto \mathbb{R}$ and let $\mathfrak{Image}_f(\mathcal{W}) = \{f(x) : x \in \mathcal{W}\} \subseteq \mathbb{R}$. In order to map the set $\mathcal{W} \times \mathfrak{Image}_f(\mathcal{W})$ to the polar coordinate system we proceed as follows.

First, we discretize $\mathcal{W}$ using a uniform grid of dimension $2^N \times 2^N$. For each point $(x_i, y_i),\ (1 \leq i \leq 2^N)$ we compute the corresponding (r_i, θ_i) point in polar form using the following equations:

$$\begin{aligned} r_i &= \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}, \\ \theta_i &= \operatorname{atan2}(y_i - y_0, x_i - x_0), \end{aligned} \tag{4.2}$$

where (x_0, y_0) is the agent's current position and (r_i, θ_i) is the distance and angle position with respect to the agent. Let $\phi_{(x_0,y_0)} : \mathbb{R}^2 \mapsto \mathbb{R} \times S^1$. We can then associate to each pair (r_i, θ_i) the function value $f(x_i, y_i)$, that is, $f'(r_i, \theta_i) = f(\phi_{(x_0,y_0)})(x_i, y_i)) = f(x_i, y_i)$.

The next step is to obtain a multiresolution (rectangular) cell approximation of the function f' in $\mathcal{W}'$. As we shall see in the next section, the wavelet transform provides us with such a multiresolution cell based approximation in any rectangular domain. Based on this cell decomposition we can proceed with the design of our path planning algorithm working entirely in the $\mathcal{W}'$ domain by applying the wavelet-based path planning algorithm of Chapter 3.

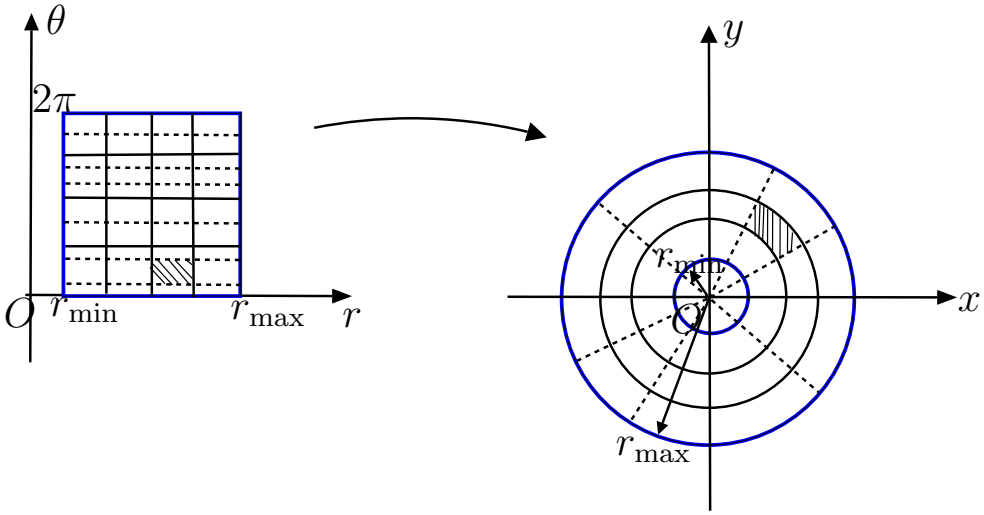


Figure 4.4: A multiresolution approximation of the rectangular domain in (r, θ) system defined by the radii $r_{\min}$ and $r_{\max}$ under the inverse conformal mapping gives a multiresolution sector approximation of an annulus cut defined by the same radii.

4.4 A Multiresolution Decomposition Scheme of the Risk Measure

Without loss of generality, in the work we take $\mathcal{W} = [0,1] \times [0,1]$. Using the conformal mapping of section 4.2, $\mathcal{W}$ is mapped to $\mathcal{D}'$ in the polar coordinate system. The new domain however, is not rectangular. Assuming without loss of generality (renormalize $\mathcal{W}$, if necessary) that $\mathcal{D}' \subset [0,1] \times [0,1]$, we let $[0,1] \times [0,1] \backslash \mathcal{D}'$ lie completely inside the obstacle configuration space $\mathcal{O}'$ in the (r, θ) domain. Thus, by adding this artificial obstacle corresponding to the boundary of $\mathcal{D}'$ we can assume that the world $\mathcal{W}'$ in the polar coordinate system to be again $\mathcal{W}' = [0,1] \times [0,1]$.

We describe $\mathcal{W}'$ using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from the previous discussion that the Haar wavelet decomposition of a function f' defined over $\mathcal{W}'$

at resolution level $J \geq J_{\min}$, given by,

$$f'(r,\theta) = \sum_{k,\ell=0}^{2^{J_{\min}}-1} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(r,\theta) + \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J-1} \sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(r,\theta) \tag{4.3}$$

induces a cell decomposition of $\mathcal{W}'$ of square cells of size $1/2^J \times 1/2^J$ in the (r,θ) coordinate system.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto [0,1]$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$. For instance, one may choose

$$\mathsf{rm}(\mathsf{x}) = \begin{cases} (d_{\max} - \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_2)/d_{\max}, & \text{if } \mathsf{x} \in \mathcal{F}, \\ 1, & \text{if } \mathsf{x} \in \mathcal{O}, \end{cases} \tag{4.4}$$

where $d_{\max} \stackrel{\triangle}{=} \max_{\mathsf{x}\in\mathcal{F}} \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_2$. Alternatively, one may think of rm as the probability that $(x,y) \in \mathcal{O}$. The function rm can be defined over $\mathcal{W}'$, i.e $\mathsf{rm}' : \mathcal{W}' \mapsto [0,1]$.

All points within range ρ from the current location of the agent, when expressed in the (r,θ) system, are given by

$$\mathcal{N}(\rho) \stackrel{\triangle}{=} \{(r,\theta) \in \mathcal{W}' : |r| \leq \rho,\ \theta \in [-\theta_{\min}, \theta_{\max}]\}. \tag{4.5}$$

The region that corresponds to $\mathcal{N}(\rho)$ in the (r,θ) coordinate system is a strip of width equal to ρ and height $\theta_{\max} - \theta_{\min}$. For simplicity, in this work we will assume that $\theta_{\min} = -\pi$ and $\theta_{\max} = \pi$.

Suppose now that we are given the desired levels of resolution of $\mathcal{W}'$ as $J_{\min} \leq j \leq J_{\max}$ where $J_{\min}, J_{\max} \in \{1,\ldots,N\}$, with corresponding ranges ρ_j from the agent's current location. By this we mean that we wish all points $(r,\theta) \in \mathcal{N}(\rho_{J_{\max}})$ to be described by resolution $J_{\max}$, all points $(r,\theta) \in \mathcal{N}(\rho_{j-1}) \backslash \mathcal{N}(\rho_j)$ to be described by resolution j, where $J_{\min} < j \leq J_{\max}$, and all points $(r,\theta) \notin \mathcal{N}(\rho_{J_{\min}+1})$ to be described by resolution $J_{\min}$. Since we require finer resolution closer to the agent we assume, of course, that $\rho_{j-1} > \rho_j$.

The choice of $J_{\max}$ is dictated by the requirement that at this level all cells should be resolved into either free or occupied cells. The choice of $J_{\min}$ as well

as the values of ρ_j are typically dictated by the sensor specifications and/or the on-board computational resources.

We obtain the distinct resolution levels at the given required distances from the current location of the agent by applying the Haar wavelet transform to rm'. To this end, let $\mathcal{I}(j) \triangleq \{0, 1, \ldots, 2^j - 1\}$ and let

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,k} \cap [0, \rho_j] \neq \varnothing\},$$
$$\mathcal{L}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,\ell} \cap [\theta_{\min}, \theta_{\max}] \neq \varnothing\}.$$

The wavelet decomposition of rm', given by

$$\begin{aligned}\mathsf{rm}'(r,\theta) = &\sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(r,\theta) \\ &+ \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k \in \mathcal{K}(j) \\ \ell \in \mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(r,\theta)\end{aligned} \tag{4.6}$$

induces, via a slight abuse of notation, the following cell decomposition on $\mathcal{W}'$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}} \tag{4.7}$$

where, ΔC_d^j is a union of square cells $c^j_{k,\ell}$ in the (r, θ) domain of dimension $1/2^j \times 1/2^j$. Furthermore, by applying the inverse mapping based on the analysis of section 4.2 we obtain a sector decomposition $\mathcal{S}_d$ of $\mathcal{W}$, where $\mathcal{S}_d$ is defined as

$$\mathcal{S}_d = \Delta S_d^{J_{\min}} \oplus \cdots \oplus \Delta S_d^{J_{\max}} \tag{4.8}$$

where, ΔS_d^J is the image under the inverse conformal mapping of ΔC_d^J, with $J_{\min} \leq J \leq J_{\max}$.

4.5 Sector-based Multiresolution Path Planning

The proposed multiresolution path planning algorithm takes place completely in the (r, θ) coordinate system. The agent in this system remains at the origin at all times. Let $\mathsf{x}' = (r, \theta)$. Assuming therefore that $\mathsf{x}' = (0, 0)$ at $t = t_0$, we construct using the approach of Section 4.4, a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}'$. Denote by $\mathcal{G}(t_0)$ be the corresponding connectivity graph. Using Dijkstra's algorithm (or any

other similar algorithm) we then find a *tentative* path $\mathcal{P}(t_0)$ in $\mathcal{G}(t_0)$ of free and mixed nodes which connects the initial node to the final node in $\mathcal{G}(t_0)$. The path $\mathcal{P}(t_0)$ is therefore an ordered sequence of nodes

$$\mathcal{P}(t_0) = (v_1^1, v_2^1, \cdots, v_{\ell_1-1}^1, v_{\ell_1}^1 = v_f^1). \tag{4.9}$$

where $v_i^1 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_i')$ and $\mathsf{x}_i' = \mathsf{cell}_{\mathcal{G}(t_0)}(v_i^1)$ is a representative, arbitrary point of the cell that corresponds to node v_i^1. For simplicity, we will assume that x_i' is the center of the cell.

The first cell in the sequence (4.9) is the cell that contains the origin and the final cell in the sequence is the cell that contains the final destination (r_f, θ_f) provided that such a sequence exists. Since we assume high resolution inside $\mathcal{N}(\rho_{J_{\max}})$ it is natural to assume that the first two cells in $\mathcal{C}_d(t_0)$ corresponding to v_1^1 and v_2^1 are free for a feasible path. The agent subsequently moves from $\mathsf{cell}_{\mathcal{G}(t_0)}(v_1^1)$ to $\mathsf{cell}_{\mathcal{G}(t_0)}(v_2^1)$. This means that only the points x_1' and x_2' will actually be visited by the agent. In order to find the subsequent points of the actual path to be followed we apply a continuous replanning scheme. This is accomplished by constructing a cell decomposition $\mathcal{C}_d(t_k)$ at each next time step $t_k > t_0,\ (k = 1, 2, \ldots)$, and by inserting in the list of the visiting points only the point $\mathsf{x}_k' = \mathsf{cell}_{\mathcal{G}(t_k)}(v_2^k)$ which corresponds to the second node of the tentative path $\mathcal{P}(t_k)$. We repeat the process until the goal is inside the second cell of $\mathcal{P}(t_k)$.

Since the approach does not eliminate the possibility that the same point may be revisited during the subsequent replanning, thus resulting in an endless loop, we overcome this issue by keeping a list of all points already visited and by removing the corresponding nodes form the graph $\mathcal{G}(t_k)$ at each time step t_k. A pseudo-code implementation of the above algorithm is given in Fig. 4.5.

```
BEGIN PATH PLANNING ALGORITHM
{
    x'_0 = 0;
    i = 0;
    x_i ⟵ x_0;
    {L_Visited} = {∅};
    (while ||x_f − x_i|| > ε)
    {
      compute rm(x, i) for all x ∈ W;
      compute rm(x', i) for all x' ∈ W' via conformal mapping;
      construct C_d(i) on W' topology ;
      construct G(i) = (E(i), V(i));
      (if L_Visited is nonempty)
        for v ∈ V(i)
          x'_v(i) = cell_G(i)(v);
          if x_v(i) ∈ L_Visited
            extract v from V(i);
            for all u adjacent to v
              remove (v, u) from E(i);
          end if;
      end if;
      v_1^i ⟵ node_G(i)(x'_0);
      v_f^i ⟵ node_G(i)(x'_f);
      P(i) ⟵ Dijkstra(v_1^i, v_f^i, V(i), E(i));
      if P(i) = {∅}
        report FAILURE;
        break;
      x'_i(1) = cell_G(i)(v_1^i);
      x'_i(2) = cell_G(i)(v_2^i);
      {L_Visited} ⟵ {L_Visited} ⨁{x'_i(1), x'_i(2)};
      x'_{i+1} ⟵ x'_i(2);
      i ⟵ (i + 1);
      compute x_{i+1} via inverse conformal mapping;
      x_0 ⟵ x_{i+1};
    }
}
END PATH PLANNING ALGORITHM
```

Figure 4.5: Pseudo-code implementation of proposed multiresolution path planning scheme.

Chapter 5

Time Scheduling and Smooth Trajectory Generation

5.1 Introduction

As we have seen in Chapters 3 and 4 the output of the proposed *path planning scheme* at each time step t_j is a path $\mathcal{P}(t_j)$. This path is defined as a finite ordered sequence of visiting points x_i, where i belongs to some index set $\mathcal{I}(t_j)$. These points form at each time t_j a polygonal line which lies completely inside the free space $\mathcal{F}$. The engineering problem that subsequently appears is to derive control laws that result in this polygonal trajectory curve. The difficulty of this problem is due to the dynamic constraints imposed by the equations of motion of the agent. These constraints make the tracking of the polygonal ,and thus non smooth, trajectory a "mission impossible". In this section we examine ways to generate smooth trajectories when the equations of motion of the agent are specified by a unicycle kinematic model (ground vehicle model). Our objective is that the generated trajectory passes sufficiently close to the x_i points at specified instants of time (time scheduling).

5.2 Unicycle Kinematic Model under Dynamic Extension

The unicycle kinematic model is given by the following equations:

$$\dot{x}(t) = v(t)\cos(\theta(t)) \tag{5.1}$$

$$\dot{y}(t) = v(t)\sin(\theta(t)) \tag{5.2}$$

$$\dot{\theta}(t) = \omega(t) \tag{5.3}$$

where x, y are the coordinates of the unicycle, θ is the orientation of the vehicle (velocity vector orientation) and v and ω are the control inputs of the system. Specifically, v is the speed of the vehicle defined by the following equation

$$v(t) = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}. \tag{5.4}$$

Furthermore, ω is the angular velocity of the agent due to the change of vehicle orientation. Since the output we wish to track is the position of the agent we choose the outputs of our system to be

$$h_1(t) = x(t) \tag{5.5}$$

$$h_2(t) = y(t) \tag{5.6}$$

The system can be written in the form $\dot{\mathsf{x}}(t) = f(\mathsf{x}) + g(\mathsf{x})u$ with

$$\mathsf{x}(t) = \begin{pmatrix} \mathsf{x}_1(t) \\ \mathsf{x}_2(t) \\ \mathsf{x}_3(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix},$$

$$f(\mathsf{x}) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad g(\mathsf{x}) = \begin{pmatrix} \cos(\mathsf{x}_3) & \sin(\mathsf{x}_3) & 0 \\ 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}, \; u(t) = \begin{pmatrix} v \\ \omega \end{pmatrix},$$

where the output is written in the form

$$y(t) = h(\mathsf{x}(t))$$

where

$$h = \begin{pmatrix} h_1(t) \\ h_2(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}.$$

Our goal then is to find the admissible inputs v, ω which achieve asymptotic tracking of the system output to a desired reference signal

$$\tilde{h}_{\text{ref}} = \begin{pmatrix} x_{\text{ref}}(t) \\ y_{\text{ref}}(t) \end{pmatrix}.$$

Since the system in (5.1)-(5.3) is nonlinear, in order to achieve asymptotic tracking our first thought is to try to apply input/output feedback linearization (i.e. IOL, see [15]). Unfortunately, the relative degrees r_1, r_2 for the specific choice of inputs and outputs are found to be ill defined. To see this, let $\mathcal{L}_\phi z(x) = \frac{\partial z}{\partial x}\phi(x)$ denote the Lie derivative of z with respect to ϕ along ϕ (where ϕ and z are sufficiently smooth functions). Then, after routine calculations we have for the first output

$$\mathcal{L}_{\text{col}_1(g)} h_1(\mathsf{x}) = \cos(\mathsf{x}_3) \tag{5.7}$$

$$\mathcal{L}_{\text{col}_2(g)} h_1(\mathsf{x}) = 0 \tag{5.8}$$

and for the second one

$$\mathcal{L}_{\text{col}_1(g)} h_2(\mathsf{x}) = \sin(\mathsf{x}_3) \tag{5.9}$$

$$\mathcal{L}_{\text{col}_2(g)} h_2(\mathsf{x}) = 0 \tag{5.10}$$

where the notation $\text{col}_1(g)$ is used for the i^{th} column of g. Therefore, the decoupling matrix corresponding to (5.1)-(5.3) is given by

$$\alpha = \begin{pmatrix} \mathcal{L}_{\text{col}_1(g)} h_1 & \mathcal{L}_{\text{col}_2(g)} h_2 \\ \mathcal{L}_{\text{col}_1(g)} h_1 & \mathcal{L}_{\text{col}_2(g)} h_2 \end{pmatrix} = \begin{pmatrix} \cos(\mathsf{x}_3) & 0 \\ \sin(\mathsf{x}_3) & 0 \end{pmatrix}. \tag{5.11}$$

where the notation

From (5.11) we observe that the matrix α is always singular and thus we cannot find a region in $\mathbb{R}^3$ where both r_1 and r_2 are non zero. Therefore, application of output tracking via input/output feedback linearization (IOL) theory is impossible.

A proposed methodology in the literature (see [15]) to overcome this problem is to achieve well-defined relative degree via dynamic extension. The idea is to transform the system dynamics in such a way that all the Lie derivatives in the expressions (5.7) - (5.10) vanish. In that case, both r_1 and r_2 are at least equal to

one. The most desirable case is to achieve $r_1 + r_2 = n$, where n is the dimension of the new system dynamics. In that case, the system is said to have trivial zero dynamics and therefore no internal instability phenomena appear. The term zero dynamics describes these modes of the system that do not allow the original dynamical system to be brought into a fully linear and controllable form via state feedback and coordinate transformation. If however, $2 \leq r_1 + r_2 < n$ then because the decoupling matrix is nonsingular our system can be separated into two subsystems in cascade form after the application of the feedback linearization theory. The first one is a controllable linear system, whereas the second subsystem is formed by the remaining zero dynamics which cannot be reached directly by the input. Explicit or implicit interconnections between these two subsystems are possible. Thus, in order to guarantee that the system does not exhibit any internal instability the remaining zero dynamics have to be stable in the sense of Lyapunov.

In our case, we proceed by considering the speed v to be an additional state of the system dynamics and

$$\tilde{v} = \frac{dv}{dt} \tag{5.12}$$

to be the new control input component in place of v, or in other words the speed v(new state) is the output of an integrator driven by the new control input $\tilde{v}$. Then, the system dynamics (5.1)-(5.3)become:

$$\dot{x}(t) = v(t)\cos(\theta(t)) \tag{5.13}$$

$$\dot{y}(t) = v(t)\sin(\theta(t)) \tag{5.14}$$

$$\dot{\theta}(t) = w(t) \tag{5.15}$$

$$\dot{v}(t) = \tilde{v}(t) \tag{5.16}$$

The system can be written in the form $\dot{\tilde{\mathsf{x}}}(t) = \tilde{f}(\tilde{\mathsf{x}}(t)) + \tilde{g}(\tilde{\mathsf{x}}(t))\tilde{u}(t)$ with

$$\tilde{\mathsf{x}}(t) = \begin{pmatrix} \tilde{\mathsf{x}}_1(t) \\ \tilde{\mathsf{x}}_2(t) \\ \tilde{\mathsf{x}}_3(t) \\ \tilde{\mathsf{x}}_4(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \end{pmatrix},$$

$$\tilde{f}(\tilde{\mathsf{x}}(t)) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \tilde{g}(\tilde{\mathsf{x}}(t)) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}^{\mathrm{T}}, \; \tilde{u}(t) = \begin{pmatrix} \tilde{v}(t) \\ w(t) \end{pmatrix},$$

where the output is as before written in the form

$$\tilde{y}(t) = \tilde{h}(\tilde{\mathsf{x}}(t))$$

where $\tilde{h} = \begin{pmatrix} \tilde{h}_1(t) \\ \tilde{h}_2(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$.

Repeating the Lie derivatives calculations for the dynamically extended system we get for the first output

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = 0 \tag{5.17}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = 0 \tag{5.18}$$

and for the second one

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = 0 \tag{5.19}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = 0 \tag{5.20}$$

so the relative degrees r_1 and r_2 are well defined and both of them greater than or equal to one. Therefore, we can continue the calculations for higher order Lie derivatives to obtain for the first output

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = \cos(\tilde{\mathsf{x}}_3)) \tag{5.21}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_1(\tilde{\mathsf{x}}) = -\tilde{\mathsf{x}_4}\sin(\tilde{\mathsf{x}}_3) \tag{5.22}$$

and for the second one

$$\mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = \sin(\tilde{\mathsf{x}_3}) \tag{5.23}$$

$$\mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2(\tilde{\mathsf{x}}) = \tilde{\mathsf{x}_4}\cos(\tilde{\mathsf{x}}_3) \tag{5.24}$$

Therefore the corresponding decoupling matrix to the dynamically extended unicycle (5.13)-(5.16) is given by

$$\tilde{\alpha} = \begin{pmatrix} \mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1 & \mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2 \\ \mathcal{L}_{\mathrm{col}_1(\tilde{g})}\tilde{h}_1 & \mathcal{L}_{\mathrm{col}_2(\tilde{g})}\tilde{h}_2 \end{pmatrix} = \begin{pmatrix} \cos(\tilde{\mathsf{x}}_3) & -\tilde{\mathsf{x}_4}\sin(\tilde{\mathsf{x}}_3) \\ \sin(\tilde{\mathsf{x}}_3) & \tilde{\mathsf{x}_4}\cos(\tilde{\mathsf{x}}_3) \end{pmatrix}. \tag{5.25}$$

We observe that the matrix $\tilde{\alpha}(\tilde{\mathsf{x}})$ is non singular for all $\tilde{\mathsf{x}} \in \mathbb{R}^4$ with $\tilde{\mathsf{x}}_4 \neq 0$ and furthermore, the relative degrees are $\tilde{r}_1 = \tilde{r}_2 = 2$. Thus, we have $r_1 + r_2 = n = 4$ and the system is input/output linearizable with trivial zero dynamics. Notice that the condition $\tilde{\mathsf{x}}_4 \neq 0$ in our specific choice of the system outputs is a smoothness requirement for the curve that the agent has to follow.

Finally, the control input vector that achieves the output tracking, provided that $\tilde{\mathsf{x}}_4 \neq 0$, is given by

$$\tilde{u}(t) = \tilde{a}^{-1}(\tilde{\nu}(t) - b(\tilde{\mathsf{x}}(t))) \tag{5.26}$$

where

$$\tilde{b}(\tilde{\mathsf{x}}(t)) = \begin{pmatrix} \mathcal{L}_{\tilde{f}}^2 \tilde{h}_1(\tilde{\mathsf{x}}(t)) \\ \mathcal{L}_{\tilde{f}}^2 \tilde{h}_2(\tilde{\mathsf{x}}(t)) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

and

$$\tilde{\nu}(t) = \begin{pmatrix} \tilde{\nu}_1(t) \\ \tilde{\nu}_2(t) \end{pmatrix}$$

with

$$\tilde{\nu}_1(t) = -k_{11}(\tilde{h}_1(\tilde{\mathsf{x}}(t)) - x_{\text{ref}}(t)) - k_{12}(\mathcal{L}_{\tilde{f}}\tilde{h}_1(\tilde{\mathsf{x}}(t)) - \dot{x}_{\text{ref}}(t)) + \ddot{x}_{\text{ref}}(t) \tag{5.27}$$

$$\tilde{\nu}_2(t) = -k_{21}(\tilde{h}_2(\tilde{\mathsf{x}}(t)) - y_{\text{ref}}(t)) - k_{22}(\mathcal{L}_{\tilde{f}}\tilde{h}_2(\tilde{\mathsf{x}}(t)) - \dot{y}_{\text{ref}}(t)) + \ddot{y}_{\text{ref}}(t). \tag{5.28}$$

In the expressions (5.27)-(5.28) $k_{11}, k_{12}, k_{21}, k_{22}$ are constants chosen such that the binomials

$$P_1(s) = s^2 + k_{12}s + k_{11} \tag{5.29}$$

$$P_2(s) = s^2 + k_{21}s + k_{22} \tag{5.30}$$

$$\tag{5.31}$$

are Hurwitz.

5.3 Reference Signal Specification

As we have described in the previous section, the output of our system is the (x, y) coordinates of the agent's position. Additionally, the path $\mathcal{P}$ of the *path planning scheme* we have presented so far is comprised of a finite number of visiting points X_i in order to reach the goal destination. Therefore, the desired system output

should be a curve $\vec{r}(\xi) = (x_{\text{ref}}(\xi), y_{\text{ref}}(\xi))$, where ξ is the curve parameter, which passes through or arbitrary close to those points. Additionally, we know that the space $\mathcal{W}$ is a polygonally connected space and thus the polygonal line that connects all the points X_i in $\mathcal{P}$ lies completely in $\mathcal{W}$. Based on this observation, the curve $\vec{r}(\xi)$ we select as the desired path of the vehicle should be a smooth curve which at the same time remains as close as possible to the original polygonal line. The smoothness requirement of the curve is imposed by the tracking scheme we are employing. Furthermore, the requirement to stay close to the polygonal lines comes from the collision avoidance constraint which requires the generated trajectory to lie at all times inside $\mathcal{W}$. The most natural way to proceed is to try to approximate the polygonal line (or part of it as we shall see later) using either a polynomial or a spline interpolation scheme. The degree of the polynomial used and the number of points lying on the polygonal line for the interpolation is correlated to the closeness of the the reference curve to the polygonal line we wish to approximate.

Designing our reference curve to be close to the polygonal line may be an overconservative approach since a trajectory which lies inside the free world $\mathcal{W}$ with more plausible characteristics (e.g. a curve with less sharp corners) may exist. However, the generation of a collision free trajectory which is at the same time compatible with the system dynamics and the admissible inputs in the framework of the path planning problems we are dealing with in this work is still an open problem. Here we tacitly assume that there exist admissible inputs v and w such that the true system response under those inputs is the reference curve

$$\vec{r}(\xi) = (x_{\text{ref}}(\xi), y_{\text{ref}}(\xi)).$$

In order to proceed, we shall employ some basic theory of differential geometry for curves (see [30]). The velocity of the agent is given by

$$\vec{V}(\xi) = \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} = \left\| \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} \right\| \vec{\mathbb{T}}(\xi) \tag{5.32}$$

where

$$\vec{\mathbb{T}}(\xi) = \frac{1}{\left\| \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} \right\|} \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi}$$

is the unit tangent vector of the curve $\vec{r}(\xi)$.

The speed of the agent is the norm of the velocity, i.e.

$$v(\xi) = \left\| \frac{\mathrm{d}\vec{r}(\xi)}{\mathrm{d}\xi} \right\| = \sqrt{\left(\frac{\mathrm{d}x(\xi)}{\mathrm{d}\xi} \right)^2 + \left(\frac{\mathrm{d}y(\xi)}{\mathrm{d}\xi} \right)^2}. \tag{5.33}$$

We can alternatively express the speed as $v = \frac{\mathrm{d}s}{\mathrm{d}\xi}$ where s is the arc length of the curve between the points $\vec{r}(\xi)$ and $\vec{r}(\xi_0)$ which is given by

$$s(\xi) = \int_{\xi_0}^{\xi} \left\| \frac{\mathrm{d}\vec{r}(z)}{\mathrm{d}z} \right\| \mathrm{d}z. \tag{5.34}$$

At this point we observe that the speed v, which appears either as one of the inputs in the original unicycle equations (5.1)-(5.3) or as one of the states in the dynamically extended model equations (5.13)-(5.16), is solely specified by the $\dot{x}, \dot{y}$ quantities. Thus, given a perfect tracking capacity scheme as the one we employ, the speed is prescribed by the reference outputs $x_{\mathrm{ref}}, y_{\mathrm{ref}}$ (or more precisely by the time derivatives of the them). In other words, after any transient phenomena have disappeared from the system response, we expect the speed to be

$$v(\xi) = \sqrt{\left(\frac{\mathrm{d}x_{\mathrm{ref}}(\xi)}{\mathrm{d}\xi} \right)^2 + \left(\frac{\mathrm{d}y_{\mathrm{ref}}(\xi)}{\mathrm{d}\xi} \right)^2}.$$

Therefore, the speed depends only on the choice of the curve parameter ξ. On the other hand, the geometric curve that passes through the points x_i of the path $\mathcal{P}$ has infinitely many different parameterizations. The question that rises in this context is how we choose the curve parameter ξ.

The way to proceed is simple. Since we expect from the vehicle not only to pass sufficiently close to the points x_i of the path $\mathcal{P}(t_j)$ but additionally, to achieve this at instants of time t_j that we specify. Therefore, the curve parametrization ξ has to specify a velocity profile compatible with this requirement. Let us assume, without loss of generality, that we wish our agent to move with constant unit speed throughout the whole route to the goal destination. In that case, the curve has to be parameterized by the natural length s (intrinsic parametrization of the curve), where the parameter s was defined in (5.34). If x_i and x_{i+1} are two successive points in $\mathcal{P}(t_j)$, then the time $t_{\mathsf{x}_i \to \mathsf{x}_{i+1}}$ required for the agent to move from x_i and x_{i+1} is given by the arc length of the resulting path

$$\Delta s_{\mathsf{x}_i \to \mathsf{x}_{i+1}} = \int_{s_i}^{s_{i+1}} \left\| \frac{\mathrm{d}\vec{r}(z)}{\mathrm{d}z} \right\| \mathrm{d}z = s_{i+1} - s_i$$

since always $\left\| \frac{\mathrm{d}\vec{r}(s)}{\mathrm{d}s} \right\| = 1$. In other words, in this case there is no distinction between time and intrinsic parametrization of the curve.

We can extend this approach in the case where we wish the agent to move with varying speed other different parts of the path. A straightforward approach is to assign a higher speed when the vehicle is inside areas of high risk measure rm and lower ones otherwise. Thus, let m be the number of the district levels of the risk measure $M_1, M_2, \ldots, M_m$ over the world $\mathcal{W}$ and let $v \in [v_{\min}, v_{\max}]$ where $v_{\min} \geq \epsilon > 0$ where the tolerance ϵ is sufficiently large to guarantee that the speed never equals zero due to the constraint that the matrix $\tilde{\alpha}$ in 5.25 is non singular. We then divide $[v_{\min}, v_{\max}]$ also to m district levels and by employing a linear model we associate to the path that lies inside the region

$$\mathcal{D}_i \triangleq \{(x, y) \in \mathcal{W} : M_{i-1} \leq \mathsf{rm}(x, y) \leq M_i\}$$

the speed value

$$v_i = v_{\min} + \left(i - \frac{3}{2}\right) \frac{v_{\max} - v_{\min}}{m - 1}. \tag{5.35}$$

The parametrization of the corresponding part of the curve lying in $\mathcal{D}_i$ is consequently

$$\xi = \frac{s}{v_i} \tag{5.36}$$

since then $\frac{\mathrm{d}s}{\mathrm{d}\xi} = v_i$.

5.4 Trajectory Generation and Time Scheduling Over a Receding Horizon

Let us assume that $\mathsf{x}(t_i)$ is the agent's current position and let $\mathcal{P}(t_i)$ be the solution path (i.e. ordered sequence of m_j distinct points x_j with $j = 1, \ldots, m_j$) to the goal destination x_f based on the environment representation of time t_i. Since $\mathsf{x}(t_i)$ is the first point in $\mathcal{P}(t_i)$, we approximate the polygonal line connecting the first three points, namely $\mathsf{x}(t_i) = \mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3$ of the path $\mathcal{P}(t_i)$ with a smooth curve $\vec{r}(\xi)$ which is the reference output of our system as described in the Section 5.3 . Then we execute the trajectory tracking scheme presented in Section 5.2 for time $t_{1\to 2}$, i.e. the time required for the agent to move from $\mathsf{x}(t_i)$ to x_2 which is given by

$$t_{1\to 2} = \int_{s_1}^{s_2} \frac{\mathrm{d}s}{v(s)}. \tag{5.37}$$

Assume that $\mathsf{x}_1 = \vec{r}(\xi_1) \in \mathcal{D}_1$ and $\mathsf{x}_2 = \vec{r}(\xi_2) \in \mathcal{D}_2$ and let $\vec{r}(\xi_{12})$ be the point corresponding to the intersection $\mathcal{D}_1 \bigcap \mathcal{D}_2 \bigcap \vec{r}(\xi)$ for $\xi \in (\xi_1, \xi_2)$. Since the agent moves for the most part with constant velocity v_1 or v_2 as long as it is inside $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, the time $t_{1\to 2}$ can be estimated by

$$t_{1\to 2} \simeq \int_{s_1}^{s_{12}} \frac{\mathrm{d}s}{v_1} + \int_{s_{12}}^{s_2} \frac{\mathrm{d}s}{v_2}. \tag{5.38}$$

and because inside $\mathcal{D}_1$ and $\mathcal{D}_2$ the s and ξ parameters are connected by the relation 5.36 the previous expression reduces to

$$t_{1\to 2} \simeq \xi_2 - \xi_1. \tag{5.39}$$

After the execution of trajectory tracking scheme over a time interval $t_{1\to 2}$ the agent's new configuration is $\mathsf{x}(t_{i+1})$ where $t_{i+1} = t_i + t_{1\to 2}$. At this time instant we compute the new path $\mathcal{P}(t_{i+1})$ using the *path planning scheme* of Chapters 3 or 4, where the point $\mathsf{x}(t_{i+1})$ is now the first point of $\mathcal{P}(t_{i+1})$. We simply repeat the procedure already described until after some finite time t_f the agent's configuration is arbitrarily close to the goal destination x_f, i.e.

$$\|\mathsf{x}(t_f) - \mathsf{x}_f\| \leq \epsilon \tag{5.40}$$

for some sufficiently small positive ϵ.

When one tries to implement this approach should be very careful since one of the assumptions of the tracking scheme we use was that the reference signal is twice differentiable. Thus, the curve parametrization should not result in discontinuous jumps of the speed (i.e. discontinuity of $\left\| \frac{\mathrm{d}r(\xi)}{\mathrm{d}\xi} \right\|$). However, the agent's transition from $\mathcal{D}_1$ to $\mathcal{D}_2$ results in a discontinuous jump on the agent's velocity. In order to eliminate this pathology induced by the curve parametrization (5.36) we propose a new parametrization for the part of the curve that corresponds to the transition from $\mathcal{D}_1$ to $\mathcal{D}_2$. This parametrization is given by

$$\xi = \frac{2s}{v_1(1 - \tanh(\frac{s-s_{12}}{\delta})) + v_2(1 + \tanh(\frac{s-s_{12}}{\delta}))}, \tag{5.41}$$

where δ is a sufficiently small positive number.

The time scheduling scheme presented in this section can be seen as an implicit way to impose input constraints. Actually, the way we approach the problem is to prescribe the speed with which the agent moves along the curve. Since perfect

tracking in realistic simulation may not be achieved as fast as we want, the time scheduling procedure may not be perfectly accurate. However, it is a design issue to achieve at least a velocity profile $v(t)$ such that for the part of the trajectory which lies inside the area $\mathcal{D}_i$ the average actual velocity over the time period $\Delta t_{\mathcal{D}_i}$ (i.e period for which the agent remains inside the area $\mathcal{D}_i$) is sufficiently close to the v_i value given by (5.35), i.e

$$\left| \frac{\int_{\Delta t_{\mathcal{D}_i}} v(t) \mathrm{d}t}{\Delta t_{\mathcal{D}_i}} - v_i \right| \leq \varepsilon_0 \tag{5.42}$$

for some sufficiently small positive ε_0.

Alternatively, in problems where the time scheduling is not an important design objective we can alternatively impose constrains on the angular velocity control input using similar ideas as those presented in this section. In particular, the angular velocity using equations (5.1)-(5.3) can be written as

$$\omega = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \tag{5.43}$$

provided that $\dot{x}^2 + \dot{y}^2 \neq 0$. Since the speed of the agent is given by (5.33) and additionally the curvature of the trajectory curve is

$$\kappa = \frac{|\ddot{y}\dot{x} - \ddot{x}\dot{y}|}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{5.44}$$

the angular velocity can be written as

$$\omega = \mathrm{sign}(\ddot{y}\dot{x} - \ddot{x}\dot{y}) \frac{\kappa}{\sqrt{v}}. \tag{5.45}$$

Therefore, the magnitude of ω is given by

$$|\omega| = \frac{\kappa}{\sqrt{v}}. \tag{5.46}$$

Because the curvature κ is an invariant of the curve under arbitrary curve parametrization ξ, the only way to impose constraints over the magnitude of the angular velocity is by finding an appropriate profile for the speed of the vehicle along the solution path. To this end, let us assume that the magnitude of the angular velocity should satisfy the following condition

$$|\omega| \leq \omega_{\max} \tag{5.47}$$

where $v \in [v_{\min}, v_{\max}]$ as in Section 5.3. Then, equations (5.46) and (5.47) imply that

$$\left(\frac{\kappa}{\omega_{\max}}\right)^2 \leq v \leq v_{\max} \tag{5.48}$$

or equivalently

$$\left(\frac{\kappa}{\omega_{\max}}\right)^4 \leq \dot{x}^2 + \dot{y}^2 \leq v_{\max}^2. \tag{5.49}$$

In case we want to incorporate the time scheduling task to the above problem formulation, then we expect the appearance of conflicts between the design objectives and the problem constraints. Therefore, the solution of such a problem requires a more complex and powerful methodology compared to the time scheduling scheme presented so far.

Chapter 6

Simulation Results

6.1 Introduction

In this chapter we present simulation results of the proposed *path planning scheme* and the smooth trajectory generation and time scheduling scheme.

For the first planning algorithm we present the results for two non-trivial scenarios. In both cases, the environment is assumed to be a square of dimension 512×512 units. Hence $N = 9$ is the finest resolution possible. For simplicity, for both scenarios only two levels of resolution have been chosen to represent the environment. Inside an area of 100×100 unit cells we employ a high resolution approximation and outside this area we employ a low resolution approximation of $\mathcal{W}$.

All the above assumptions hold in the second planning scheme. The only difference here is that the high resolution area corresponds to the area inside a disk of radius 100 pixels. This is the area of 100×512 unit cells in the (r, θ) plane when it is mapped to the (x, y) plane. Outside this area we employ a low resolution approximation of $\mathcal{W}'$.

Simulation for the smooth trajectory and time scheduling scheme are presented for all the scenarios of the two algorithms.

6.2 Simulation Results of the First Scenario for the First Path Planning Scheme

In the first scenario, the environment $\mathcal{W}$ is an actual topographic (elevation) map of a certain US state with fractal-like characteristics, shown in Fig. 6.1. The blue color in this figure corresponds to areas of obstacles whereas the initial configuration of the agent is denoted by A and the desired final configuration is denoted by B. The objective is for the agent (e.g., a UAV) to follow a path from A to B while flying as low as possible, and below a certain elevation threshold. Areas with bright colors in Fig. 6.1 correspond to areas of low risk (elevation in this case) and darker colors correspond to areas of high risk (elevation in this case) that should be avoided. Solving the path-planning problem on-line at this resolution is computationally prohibitive.

In order to apply the proposed multiresolution scheme we choose five distinct risk measure levels $M_1, \ldots, M_5$, equally spaced between 0 and 1. The two levels with the highest values ($M_4 = 0.75$ and $M_5 = 1$) denote the obstacle space; that is, $m_2 = 0.75$. The rest three levels $M_1, \ldots, M_3$ denote feasible states. Level M_1 represents the unoccupied, most desirable states and we thus chose $m_1 = M_1$.

The results from the multiresolution path-planning algorithm using a fine resolution level $J_{\max} = 5$, and a low resolution at level $J_{\min} = 3$ are shown in Fig. 6.2. Specifically, Fig. 6.2 shows the evolution of the path at different time steps as the agent moves to the final destination. Figure 6.2(a) shows the agent's position at time step $t = t_{15}$ along with the best proposed path to the final destination at that time. Similarly, Fig. 6.2(b) shows the agent's position at time step $t = t_{50}$ along with the best proposed path to the final destination at that time. As seen in Fig. 6.2(c), the actual path followed by the agent differs significantly from the one predicted in either Figs. 6.2(a) or 6.2(b). This happens because at time t_{15} and t_{50} the agent does not have complete information outside the high resolution zone, and the predicted path actually penetrates the obstacle space $\mathcal{O}$. At time t_{50}, for example, the agent – being far from any obstacle – fails to anticipate the upcoming collision. As the agent gets closer to the obstacle however, and new information is gathered, the existence of the obstacle forces the agent to redirect its path. The agent reaches the final destination x_f in a collision free manner, as

seen in Fig. 6.2(c). The actual path followed lies inside areas with a low elevation level, which verifies the optimal nature of the path.

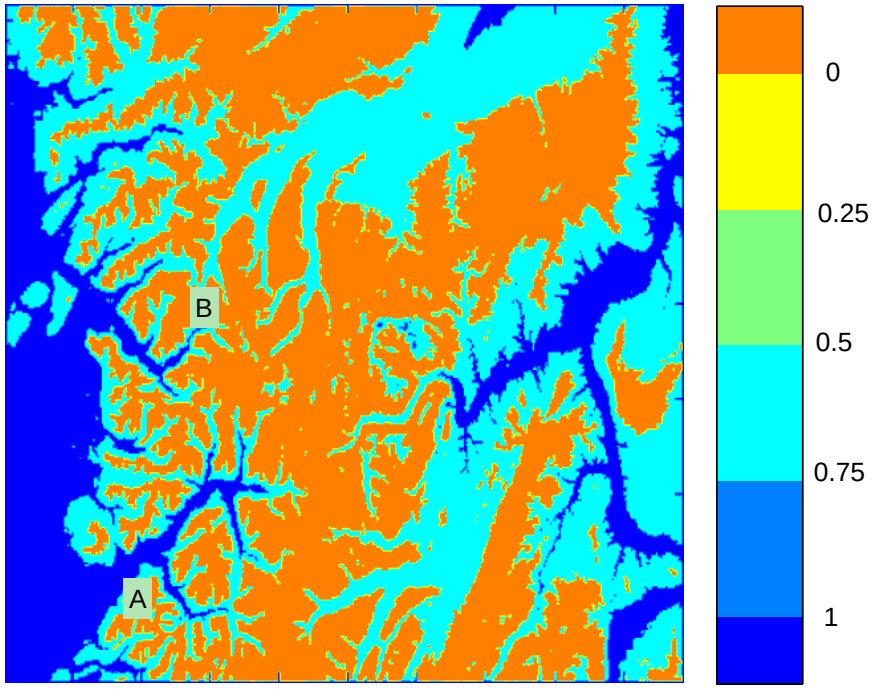
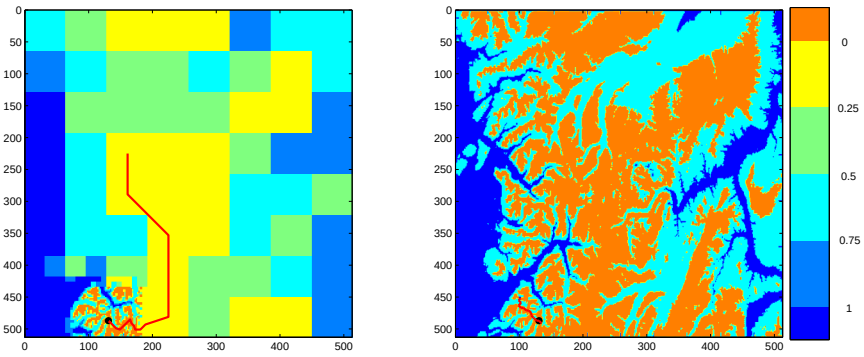
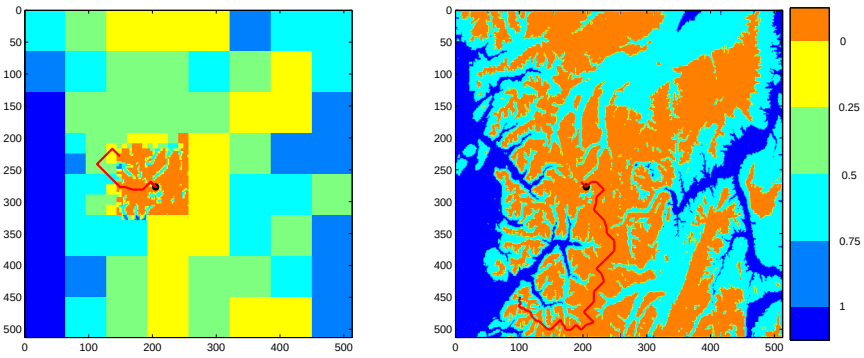


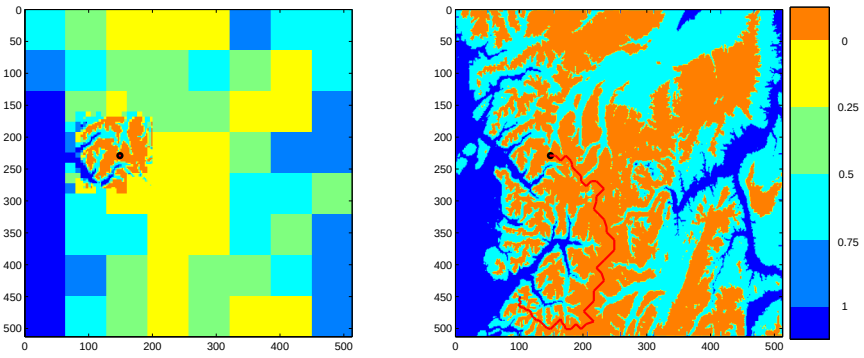
Figure 6.1: Plot of risk measure (elevation) for the whole configuration space using a 512×512 unit cell resolution.



(a) $t = 15$



(b) $t = 50$



(c) $t = 62$

Figure 6.2: Path evolution and replanning at time $t = t_{15}$, $t = t_{50}$ and $t = t_f$.

6.3 Simulation Results of the Second Scenario for the First Path Planning Scheme

In the previous scenario the cost to be minimized along the path is derived solely from the risk measure shown in Fig. 6.1. When the environment is very fragmented, this cost may result in excessively long, meandering paths. To avoid this problem for a cluttered environment as the one shown in Fig. 6.3 we add an additional term that also penalizes the total length of the path. This allows the agent to prefer short paths in the euclidean sense. Four distinct risk measure levels M_1, M_2, M_3, M_4 are chosen for the scenario shown in Fig. 6.3, as follows $M_1 = 0, M_2 = 0.25, M_3 = 0.75$, and $M_4 = 1$. Here we let again $M_2 = 0.75$ and $M_1 = 0$. Hence values above 0.75 denote obstacles, shown in red color in Fig. 6.3. The final path obtained using the proposed multiresolution path-planning algorithm is shown in Fig. 6.4.

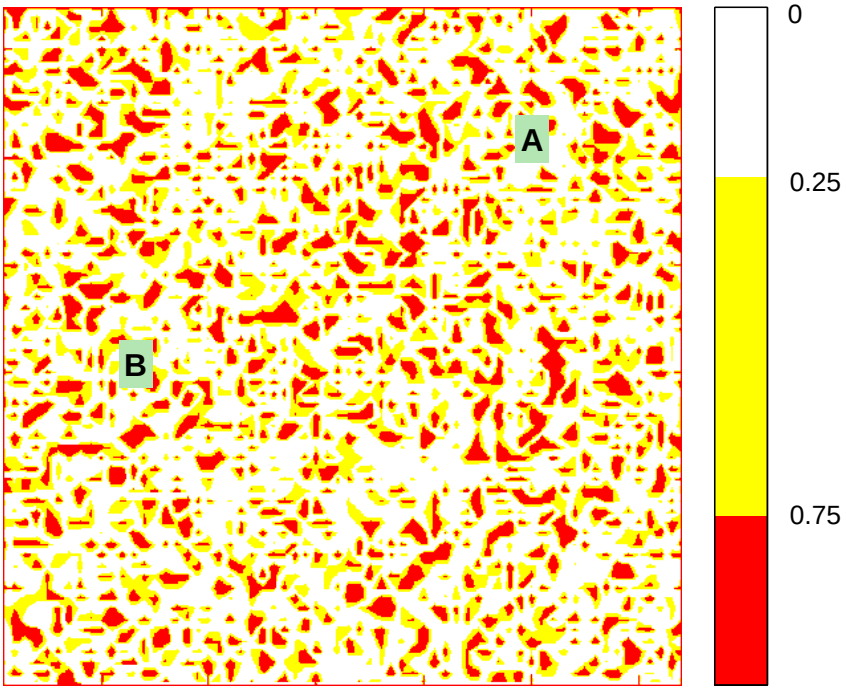


Figure 6.3: Plot of the risk measure function for the second scenario. Areas with red color correspond to the obstacle space. The point 'A' denotes the initial state x_0 and point 'B' denotes the final state x_f.

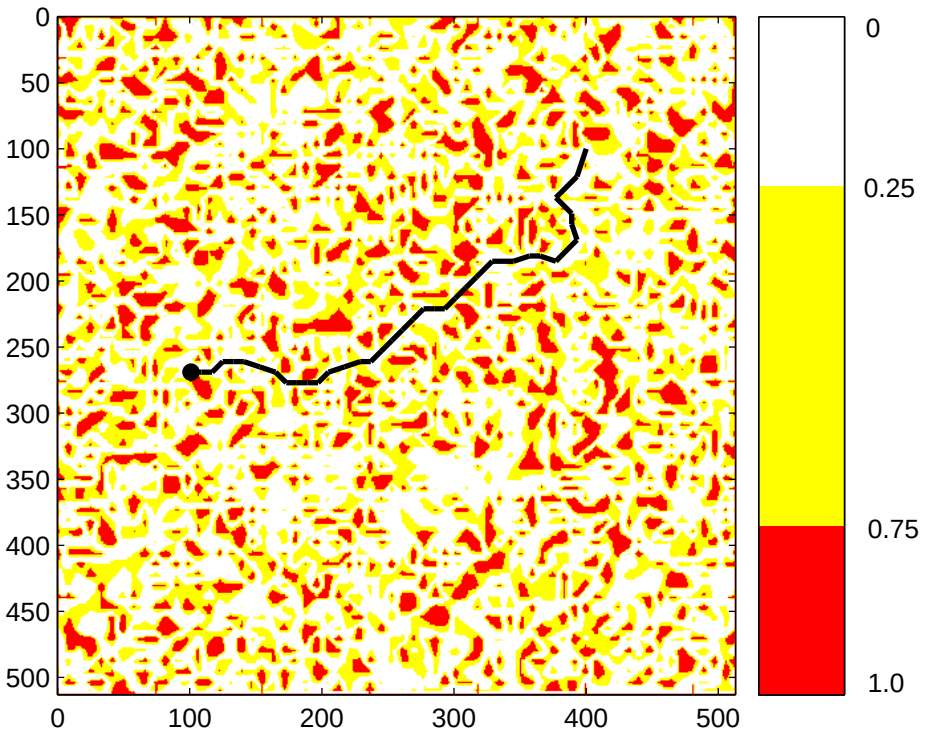


Figure 6.4: Final path for the second scenario. For such highly fragmented environments it is advisable to also include a penalty on the euclidean distance between successive nodes of the path.

6.4 Simulation Results of the Second Path Planning Scheme

In this section we present simulation results of the second proposed algorithm for a non-trivial scenario. The environment is assumed to be square of dimension 512×512 units. Hence $N = 9$ is the finest resolution possible. For simplicity, only two levels of resolution have been chosen to represent the environment. Inside an area of 100×100 unit cells in the (r, θ) system we employ a high resolution approximation and outside this area we employ a low resolution approximation of $\mathcal{W}'$.

The initial and final positions of the agent are also shown in this figure. The objective is for the agent (e.g., a UAV) to follow a path from A to B while flying as low as possible, and below a certain elevation threshold. Areas with bright colors in Fig. 6.6 correspond to areas of low risk (elevation in this case) and darker colors correspond to areas of high risk (elevation in this case) that should be avoided. Solving the path-planning problem on-line at this resolution is computationally prohibitive.

In order to apply the proposed multiresolution scheme we choose six distinct risk measure levels $M_1, \ldots, M_6$, spaced between 0 and 1. The two levels with the highest values ($M_5 = 0.9$ and $M_6 = 1$) denote the obstacle space; that is, $m_2 = 0.9$. The rest four levels $M_1, \ldots, M_4$ denote feasible states. Level M_1 represents the unoccupied, most desirable states and we thus chose $m_1 = M_1$.

The results from the multiresolution path-planning algorithm using a fine resolution level $J_{\max} = 5$, and a low resolution at level $J_{\min} = 3$ are shown in Fig. 6.5. Specifically, Fig. 6.6 shows the evolution of the path at different time steps as the agent moves to the final destination.

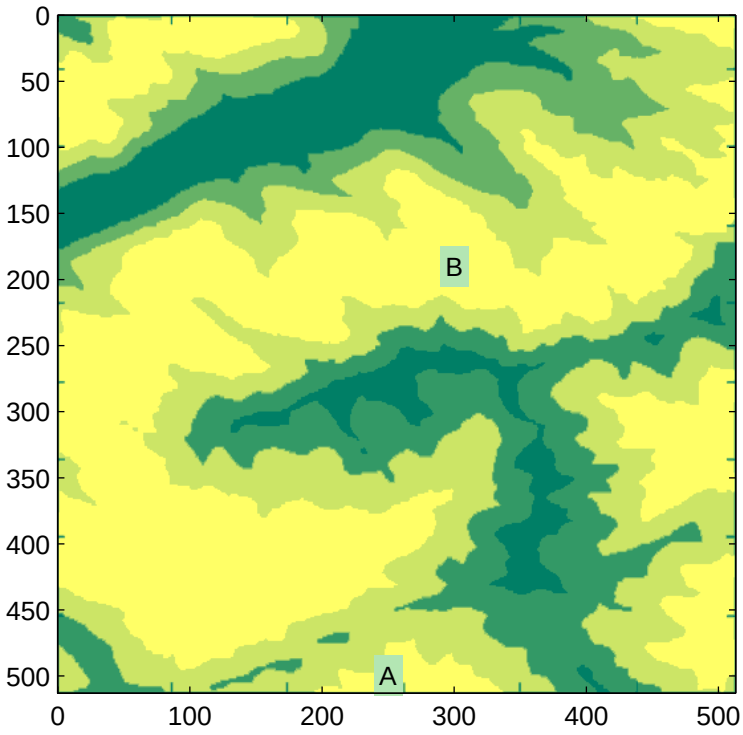
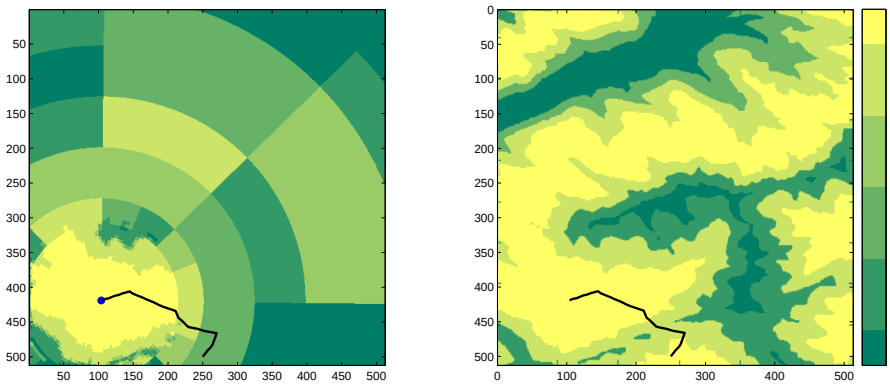
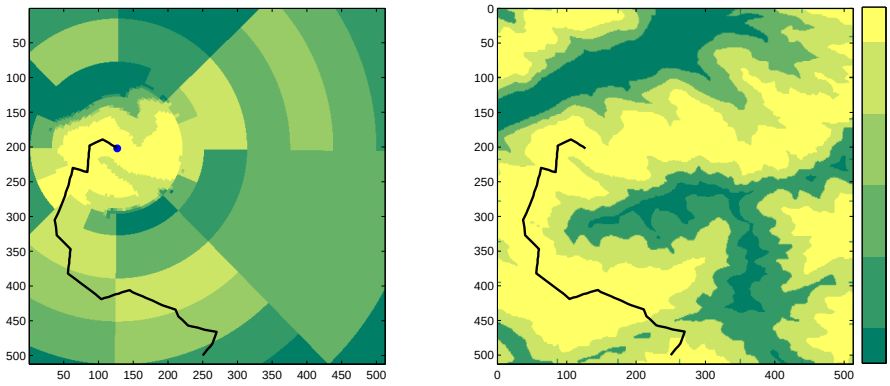


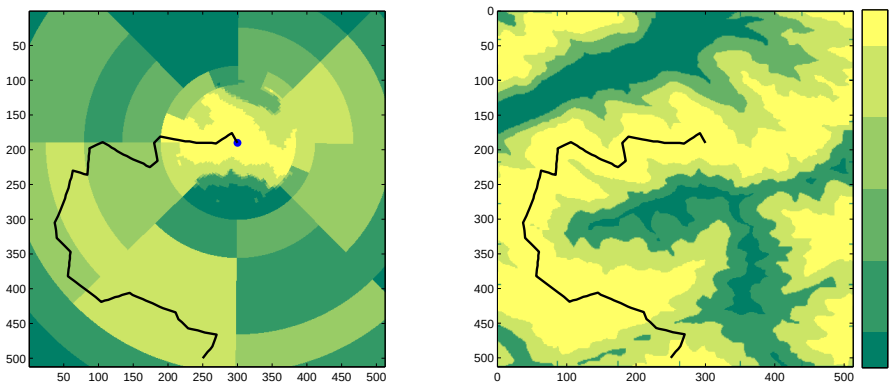
Figure 6.5: Plot of the original risk measure function where dark green area correspond to areas of high risk whereas the yellow ones to low risk areas.



(a) $t = 10$



(b) $t = 22$



(c) $t = t_f$

Figure 6.6: Path evolution. Figures show the actual path followed by the agent which finally reaches the final destination at $t = t_f$.

6.5 Simulation Results of Trajectory and Time Scheduling Scheme

In this section we present the results of the trajectory generation and scheduling scheme we presented in Chapter 5. Actually, we shall reproduce the results of the simulation for all the scenarios of the two proposed path planning algorithms, presented in the previous two sections, using the kinematic unicycle model.

To simplify the computations we require in all cases the vehicle to move with constant unit speed as long as it is inside the free world $\mathcal{F}$. The reference output signals (reference curve) correspond to a smooth approximation of the polygonal line of the path $\mathcal{P}$. In Figures 6.7-6.9 we see the simulation results on the first scenario of the first planning scheme. In Figures 6.10-6.12 we see the simulation results on the scenario with the environment full of obstacles of the first planning scheme. Finally, in Figures 6.13-6.15 the simulation results of the trajectory generation scheme for the scenario of the second planning scheme are presented.

We observe that the generated trajectory in all cases achieves the closeness requirement to the final path $\mathcal{P}$ (polygonal line) that the planning schemes give us. This guarantees that the resulting smooth curve lies completely inside the free world $\mathcal{F}$. Additionally, the trajectory at the specified time instants t_j (time scheduling) passes sufficiently close to the visiting points x_i of the corresponding available path $\mathcal{P}(t_j)$. Furthermore, the implicit constraint on the agent's speed is achieved in a satisfactory degree in the sense of condition (5.42) with ε_0 be in the order of 10^{-3}. Any deviation or oscillation phenomena that appear are due to the transient response of the system during the error regulation procedure. These are mostly due to the *on-line* implementation nature of our scheme since a small deviation in the initial conditions (especially the ones for the orientation θ) from those that correspond to the reference curve may result in a transient response for each time we execute the trajectory generation scheme. This transient response is decaying very fast and it is a design issue to make it practically disappear.

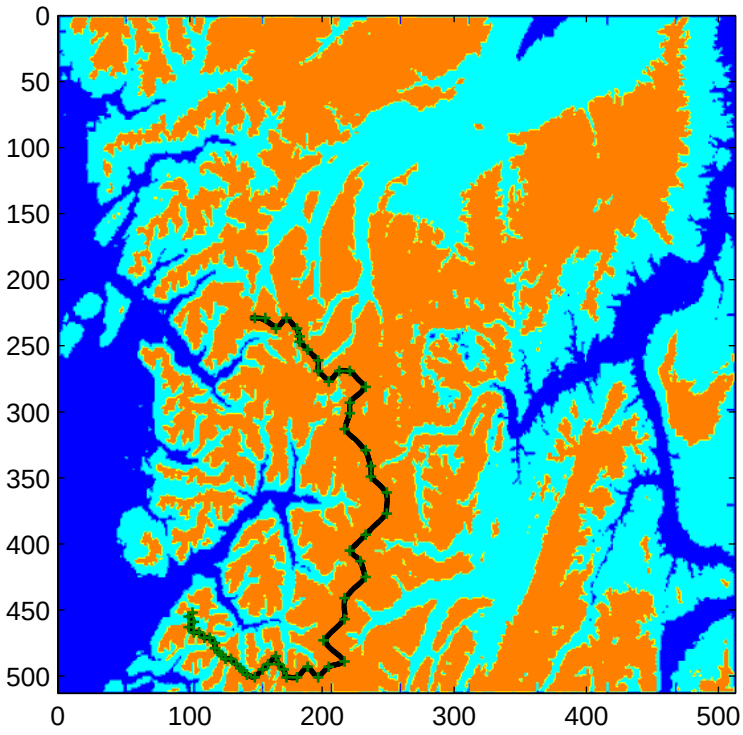


Figure 6.7: Plot of the *on-line* generated smooth trajectory passing close to the x_i points of the corresponding path $\mathcal{P}(t_j)$ generated by the first planning scheme (denoted with '+') at specified instants of time t_j for the 1^{st} scenario.

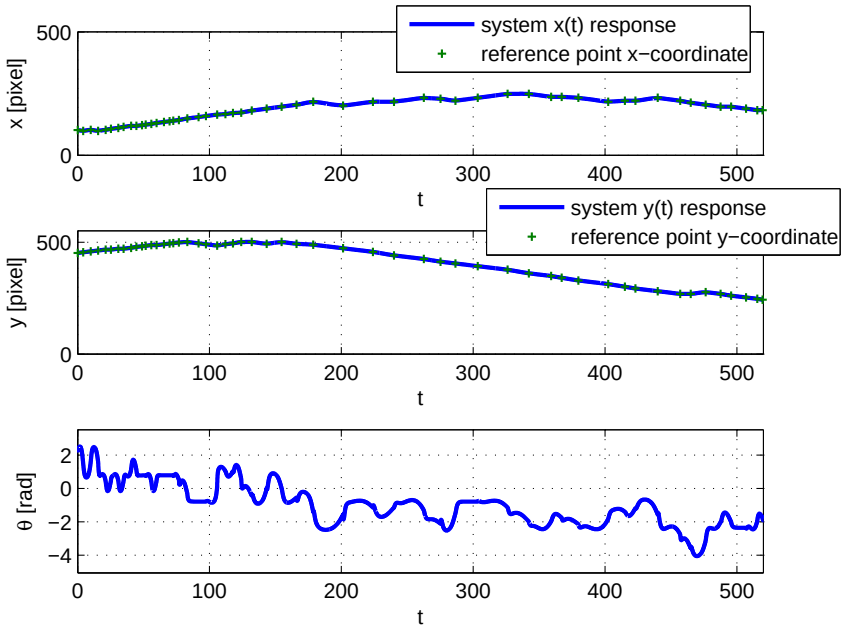


Figure 6.8: Plot of the states evolution of the system under the feedback law.

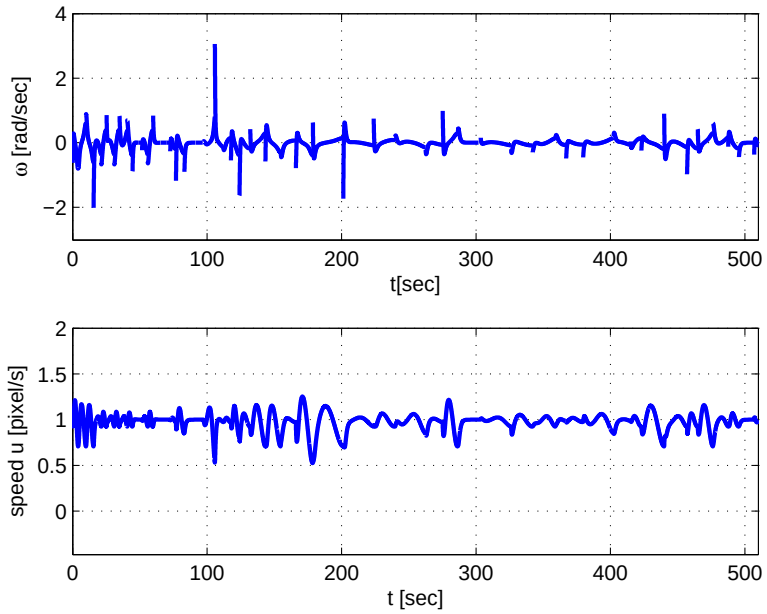


Figure 6.9: Plot of the input components (i.e. velocity v and angular velocity ω) versus time.

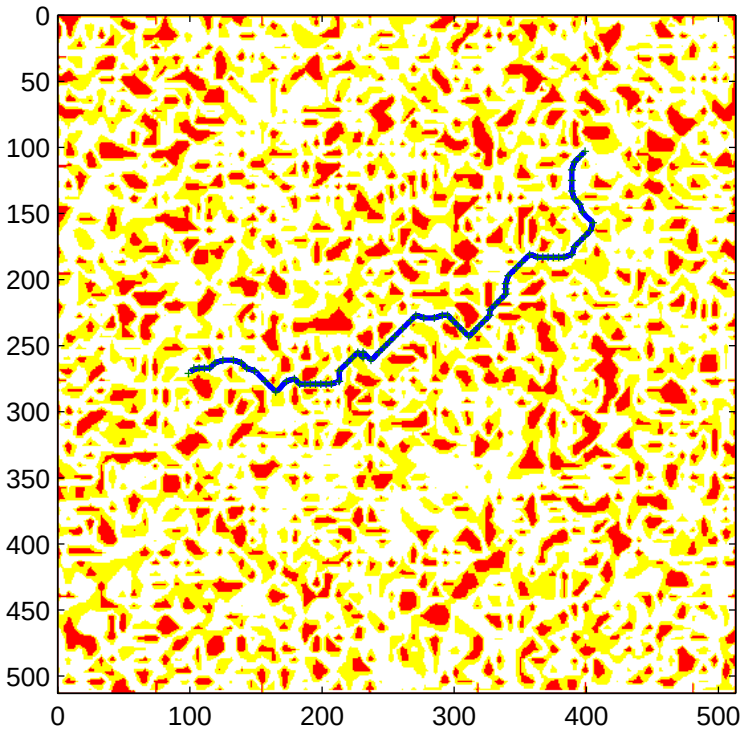


Figure 6.10: Plot of the *on-line* generated smooth trajectory passing close to the x_i points of the corresponding path $\mathcal{P}(t_j)$ generated by the planning scheme (denoted with '+') at specified instants of time t_j for the fragmented environment scenario.

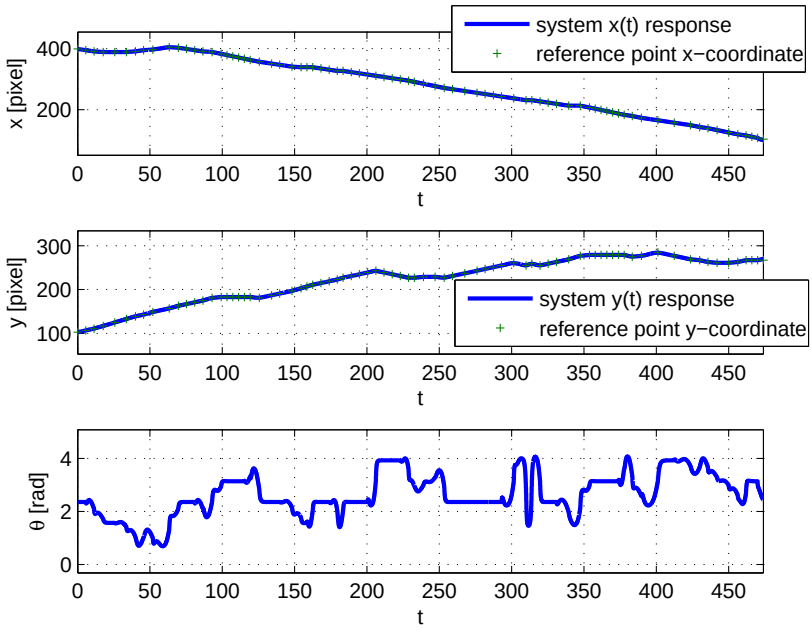


Figure 6.11: Plot of the states evolution of the system under the feedback law.

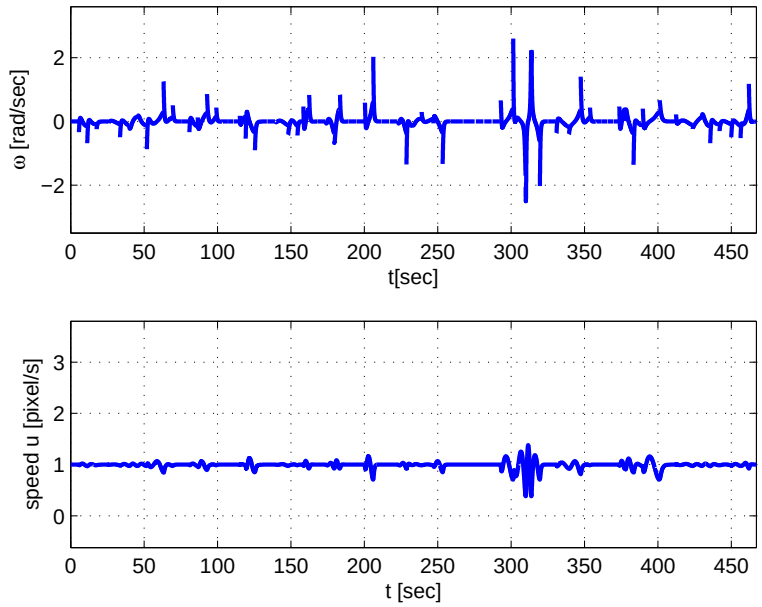


Figure 6.12: Plot of the input components (i.e. velocity v and angular velocity ω) versus time.

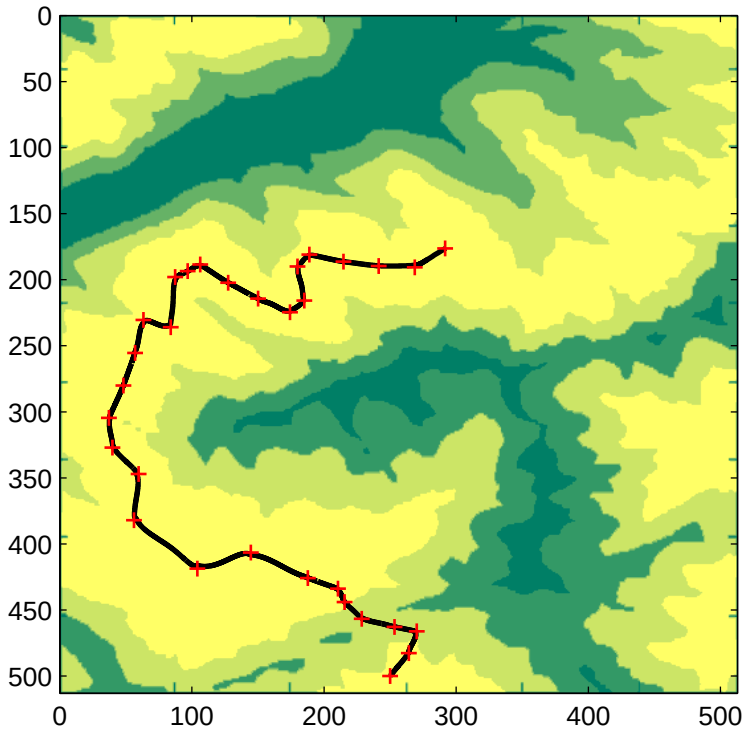


Figure 6.13: Plot of the *on-line* generated smooth trajectory passing close to the x_i points of the corresponding path $\mathcal{P}(t_j)$ generated by the second planning scheme (denoted with '+') at specified instants of time t_j.

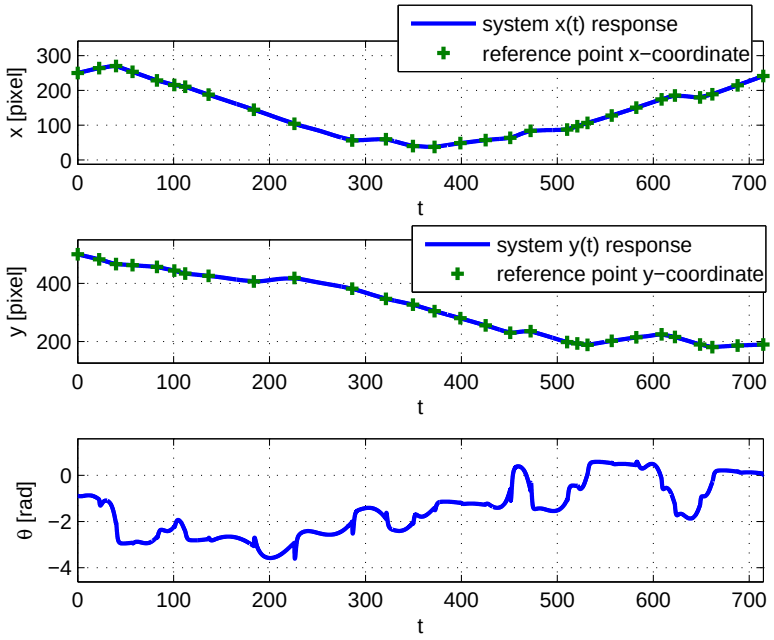


Figure 6.14: Plot of the states evolution of the system under the feedback law.

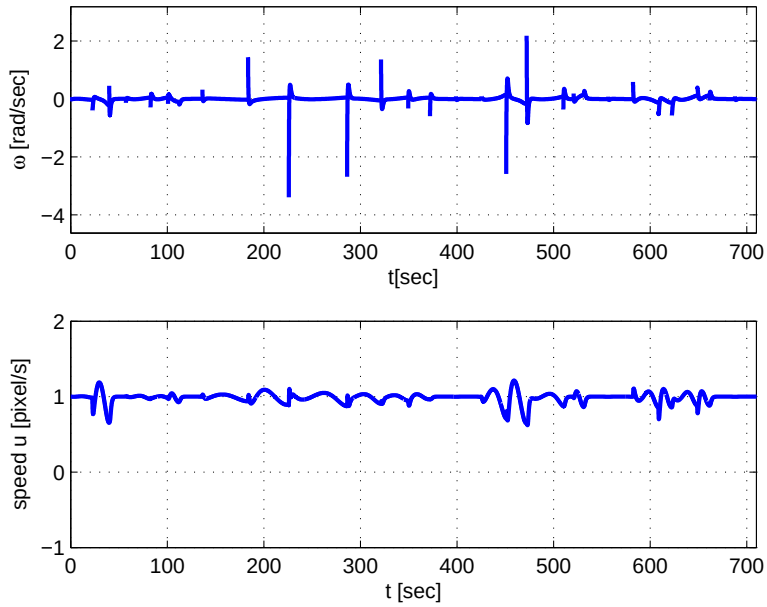


Figure 6.15: Plot of the input components (i.e. velocity v and angular velocity ω) versus time.

Chapter 7

Conclusions and Future Work

In this work we have proposed an *on-line* hierarchical *path planning scheme* for navigating an autonomous agent inside an unknown environment based on information obtained by its available on-board sensor devices. The idea is to use a higher resolution close to the agent where is needed most, and a coarser resolution at large distances from the current location of the agent. This is motivated by the natural observation that for on-line implementations it is not prudent from a computational point of view to compute a solution with great accuracy over large ranges of over a very long time horizon. The *planning scheme* is scalable and can be tailored to the available computational resources of the agent. An extension of the original idea motivated by practical consideration is also presented. In this extension, a sector-based multiresolution decomposition of the environment is computed at each step. This decomposition is adapted to the on-board sensor data using the wavelet transform in conjunction with a conformal mapping to new (polar) coordinates. The multiresolution approach allows the agent to blend information arising from different sensors at different ranges and resolutions. Furthermore, some dynamical considerations for the agent comes into play by employing a kinematic unicycle model. In order to produce smooth trajectories compatible with the dynamic constraints we introduce a smooth trajectory generation and time scheduling scheme using some basic ideas of IOL theory for MIMO systems.

The proposed methodology can be further expanded towards different directions. One important issue is to investigate how it is possible to reduce the

computations required to obtain a solution at a given instant of time if previous information is used appropriately. Additionally, the problem of smooth trajectory generation and time scheduling can be examined for more complicated kinematic or even dynamic models for the agent under control input constraints (especially for the angular velocity). In order to obtain a solution for the constrained problem the range and resolution of the information obtained by the agent's sensors may have to be appropriately modified. In this case, the robustness of the proposed scheme under these possible modifications is another crucial issue which is strongly correlated with how efficient the path planning scheme is for real time implementation. These possible extensions of the baseline methodology presented in this thesis will be addressed in the future along with the exploration of other research horizons in the framework of the path planning problem for autonomous vehicles.

Bibliography

[1] Y. K. Hwang and N. Ahuga, "Gross motion-planning–a survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219–291, 1992.

[2] M. Athans and P. Falb, *Optimal Control, An Introduction to the Theory and Its Application*. New York: Dover, 2007.

[3] R. Bartle, *The Elements of Real Analysis*. New York: Wiley Sons Inc., second ed., 1976.

[4] H. Khalil, *Nonlinear Systems*. New Jersey: Prentice Hall, third ed., 2002.

[5] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[6] S. Lavalle, *Planning Algorithms*. New York, NY: Cambridge University Press, 2006.

[7] D. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *IEEE Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1–6, 1987.

[8] D. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," in *Advances Appl. Math.*, vol. 11, pp. 412–442, 1990.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. McGraw Hill and MIT Press, second ed., 2001.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," in *IEEE Transactions on Systems Science and Cybernetics SSC4)*, pp. 100– 107, 1968.

[11] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IEEE Proceedings of the IEEE International Conference of Robotics and Automation*, pp. 3310–3317, 1994.

[12] A. Ferguson, D. Stentz, "The delayed d* algorithm for efficient path replanning," in *IEEE Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2045– 2050, 2005.

[13] M. Koenig, S. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *IEEE Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 968– 975, 2002.

[14] S. Murphey, S. Uryasev, and M. Zabarankin, "Trajectory optimization in a threat environment," *Research Report 2003-9*, pp. 22–45, 2003.

[15] A. Isidori, *Nonlinear Control Systems*. Berlin: Springer - Verlang, third ed., 1995.

[16] I. Daubechies and W. Sweldens, "Factoring wavelets transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, 1998.

[17] T. Lozano-Perez and M. Wesley, "Automatic planning for planning collision-free paths among polyhedral obstacles," in *IEEE Trans. Syst., Man, Cybern.*, vol. 11, pp. 681–698, 1981.

[18] D. Pai and L. Reissell, "Multiresolution rough terrain motion planning," in *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 19–33, 1998.

[19] S. Kambhampati and L. Davis, "Multiresolution path planning for mobile robots planning for mobile robots," *IEEE Journal of Robotics and Automation*, pp. 135–145, 1986.

[20] S. Behnke, "Local multiresolution path planning," *Lecture Notes in Computer Science*, 2004.

[21] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Cambridge, MA: Athena Scientific, 1995.

[22] B. Bollobas, *Modern Graph Theory*. New York: Springer, 1998.

[23] C. Godsil and G. Royle, *Algebraic Graph Theory*. New York: Springer, 2001.

[24] R. Diestel, *Graph Theory*. New York: Springer, third edition ed., 2006.

[25] C. Burrus, R. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms*. New Jersey, NJ: Prentice Hall, 1998.

[26] S. Mallat, "A theory for multiresolution signal decomposition, the wavelet representation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, 1989.

[27] A. Cohen, *Numerical Analysis of Wavelet Methods*, vol. 32. Amsterdam: Elsever Science, 2003.

[28] D. Walnut, *An Introduction to Wavelet Analysis*, vol. 32. Boston: Birkhauser, 2003.

[29] P. Tsiotras and E. Bakolas, "A hierarchical on-line path-planning scheme using wavelets," in *European Control Conference*, (Kos, Greece), July 2–5 2007.

[30] M. Do Carmo, *Differential Geometry of Curves and Surfaces*. New Jersey: Prentice Hall, 1976.